NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ADAPTIVE NOISE CANCELLATION APPLIED TO
THE NUWES TEST RANGES

by

Dale W. Herdegen

December 1991

Thesis Advisor: Murali Tummala

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): EC	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification): ADAPTIVE NOISE CANCELLATION APPLIED TO THE NUWES TEST RANGES

12 PERSONAL AUTHOR(S): HERDEGEN, Dale

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1991 December	80

16 SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	adaptive filtering; noise cancellation; NUWES

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis investigates the application of adaptive filtering at the NUWES test ranges. Two adaptive algorithms, least-mean-squares and recursive-least-squares are studies. To facilitate the investigation, a model of the test ranges was developed. This model accounts for spherical spreading and linear attenuation of the propagated acoustic signals as well as the effects of doppler shift, multipath, and finite propagation delay time. After describing the model, the adaptive filtering algorithms are briefing explained. The two schemes of adaptive filtering, adaptive noise cancellation and adaptive line enhancement, are applied to the model. Simulation results of the noise cancellation and line enhacement schemes are presented for several scenarios.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL: TUMMALA, Murali	22b TELEPHONE (Include Area Code): 408-646-2645	22c OFFICE SYMBOL: EC/Tu

DD Form 1473, JUN 86 — Previous editions are obsolete. S/N 0102-LF-014-6603

Approved for public release; distribution is unlimited

ADAPTIVE NOISE CANCELLATION APPLIED TO THE NUWES TEST RANGES

by

Dale W. Herdegen
Captain, United States Marine Corps
BS, University of Minnesota, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1991

Computer Engineering

ABSTRACT

This thesis investigates the application of adaptive filtering at the NUWES test ranges. Two adaptive algorithms, least-mean-squares and recursive-least-squares are studied. To facilitate the investigation, a model of the test ranges was developed. This model accounts for spherical spreading and linear attenuation of the propagated acoustic signals as well as the effects of doppler shift, multipath, and finite propagation delay time. After describing the model, the adaptive filtering algorithms are briefly explained. Then, two schemes of adaptive filtering, adaptive noise cancellation and adaptive line enhancement, are applied to the model. Simulation results of the noise cancellation and line enhancement schemes are presented for several scenarios.

TABLE OF CONTENTS

I. INTRODUCTION

A. THE NUWES TEST RANGES

The NUWES ranges are testing and evaluation facilities for the Navy. They are comprised of a series of computer linked underwater hydrophone arrays. The hydrophone arrays, in turn, provide tracking of test vehicles launched from known locations. The tracking signals emitted by the test vehicles are received by the hydrophones and processed by the computers. The recovered signal provides identification and telemetry information.

As part of the scenario, a broadband countermeasure produces noise that disrupts the tracking signals. This interference reduces the test ranges' ability to recover vehicle identification and telemetry information.

In an effort to lessen the effect of the countermeasure on the telemetry, some form of signal recovery is necessary. This thesis investigates one type of signal recovery called adaptive filtering.

B. PROCEDURE

In this thesis, an appropriate model of the NUWES test ranges is first developed. This model accounts for signal propagation loss, wideband noise sources, propagation signal delay, signal multipath, doppler shift, and the frequency

response of the hydrophones. Next, a brief explanation of the recursive algorithms is given after which, the algorithms are developed and applied to the model. Finally, detailed simulation results of the tests conducted on several NUWES test range scenarios are presented.

C. ORGANIZATION

The thesis contains five chapters and three appendices. Chapter II develops the physical model of the NUWES test range as well as the signal model used in the simulation. Chapter III gives a brief explanation of the LMS and RLS algorithms. Chapter IV develops the application of adaptive algorithms to the model developed in Chapter II. Conclusions and recommendations on the use of the algorithms at the NUWES test ranges are presented in Chapter V. Three appendices are also provided. Appendix A contains additional results that supplement the discussion in Chapter IV. Appendix B furnishes a flow chart of the major programs used in the simulation. Appendix C contains a listing of the programs mentioned in Appendix B.

II. SIMULATION MODEL FOR THE NUWES TEST RANGE

A. BASIC SETUP

The simulation model is based upon the NUWES test ranges. The ranges consist of numerous hydrophone arrays arranged on the ocean floor. A typical layout of one of the ranges is shown in Figure 1. The circular areas indicate the regions within which the hydrophone arrays can reliably receive a signal. The area covered by the reception circles surrounding the hydrophones delineates the test range.

An acoustic source emitting a 75 kHz BPSK tracking signal is attached to each test vehicle. The tracking signal is received by the hydrophones and processed by station computers to produce tracking data.

In a noiseless environment, all test vehicles can easily be tracked; however, the addition of countermeasure noise interferes with the tracking signal and corrupts the tracking data.

To recover the corrupted tracking data, some type of filtering is needed. This thesis investigates the use of adaptive filtering to enhance the tracking of test vehicles in the presence of countermeasure noise.

To aid the investigation and analysis of signal recovery, a model of the NUWES test ranges was developed. This model

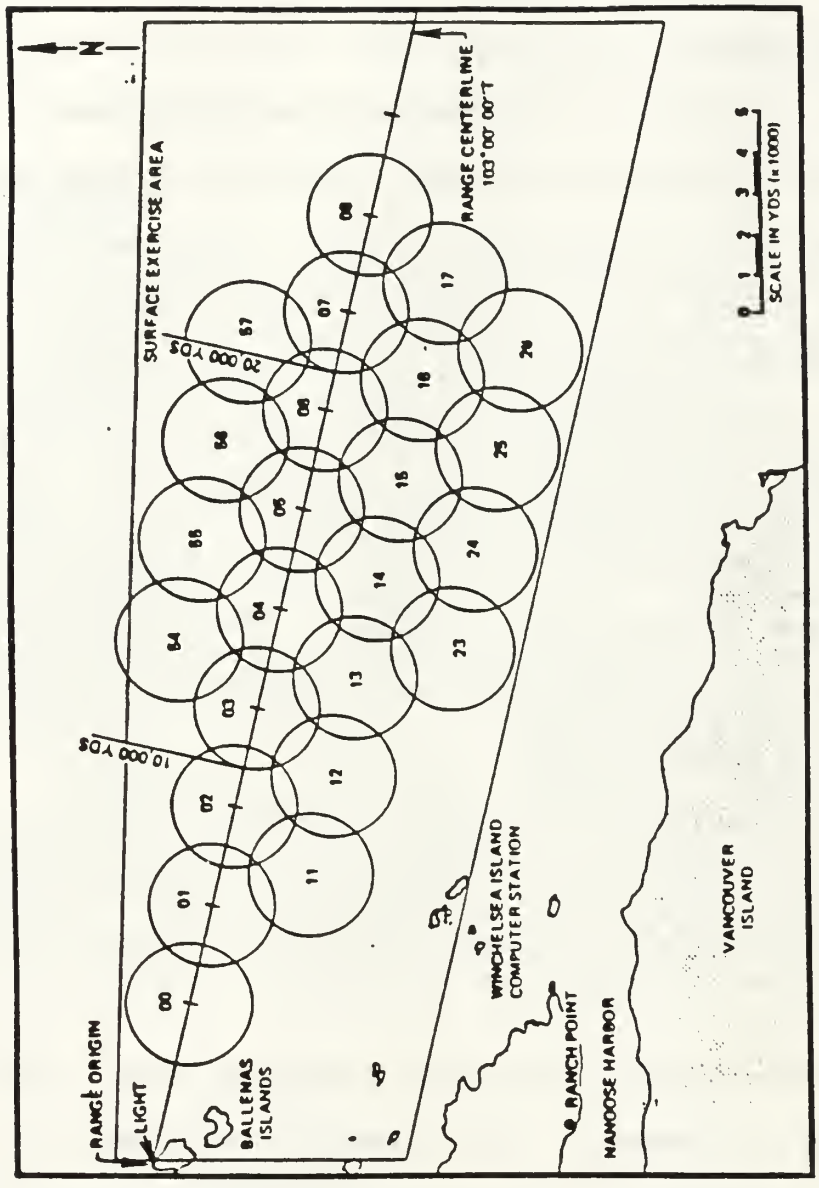


Figure 1. A hydrophone arrangement typical of the NUWES test ranges.

accounts for:

- Tracking signals
- Wideband countermeasure noise
- Spherical spreading
- Linear attenuation
- Multipath effects from surface reflection
- Doppler shift for fast moving targets

B. HYDROPHONE GRIDS

The model contains four subarrays arranged in a pattern typical of the NUWES test ranges. With these four subarrays and their attached hydrophones, any scenario in which countermeasure noise is a factor can be examined.

Figure 2 displays a mesh plot of the subarray layout. Each hydrophone subarray has a maximum effective range of 1500 meters. The range of coverage of each array is depicted by the raised circular portion of the mesh plot. The distance between hydrophone subarrays is 2500 meters. The hydrophone subarrays themselves are depicted by the small protrusion in the center of each circle.

The hydrophones and associated preamplifiers at each array are physical devices with finite frequency response characteristics. In the simulation, the frequency response of the hydrophone/preamp combination was modeled by a twelfth order Butterworth bandpass filter. The filter's magnitude and

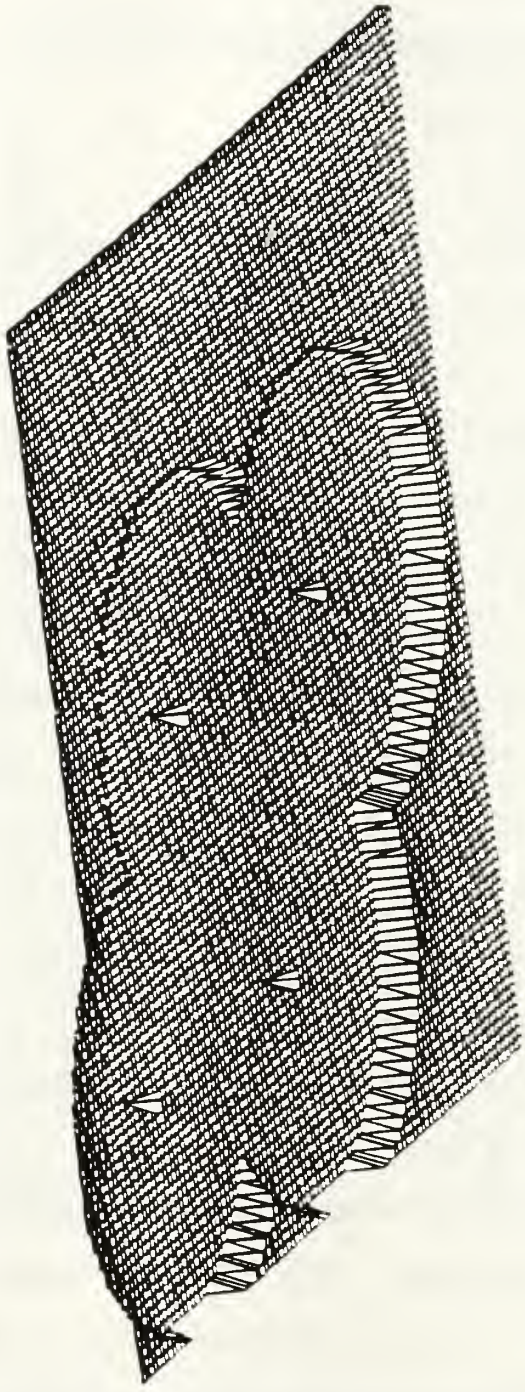


Figure 2. Mesh plot of the subarray layout.

phase response are illustrated in Figure 3. These characteristics closely match those of the actual system.

C. SIGNAL MODEL

The tracking signal is modeled as a 75 kHz BPSK signal with a code length of 47 bits. Each bit lasts 93 μs, resulting in a signal bandwidth of approximately 20 kHz. The countermeasure is a broadband jammer modeled as white noise. The signals are sampled at 300 kHz at the output of each hydrophone. The tracking signal is

$$x(n) = A\cos(2\pi f_o t) \qquad (2\text{-}1)$$

where f_0 is the carrier frequency (75 kHz) and A is a square wave of fixed magnitude representing the binary code, whose sign is positive for a 'one' bit, and negative for a 'zero' bit.

1. Signal Attenuation

Signals lose power as they propagate through the water. The transmission loss may be considered to be the sum of two types of losses: spreading and attenuation. Spreading loss accounts for the weakening of a signal as it spreads out from a source. Attenuation loss accounts for absorption and scattering. Each of these losses in signal power is proportional to the distance traveled before reception at a hydrophone. (Urick, 1983, pp. 99-103)

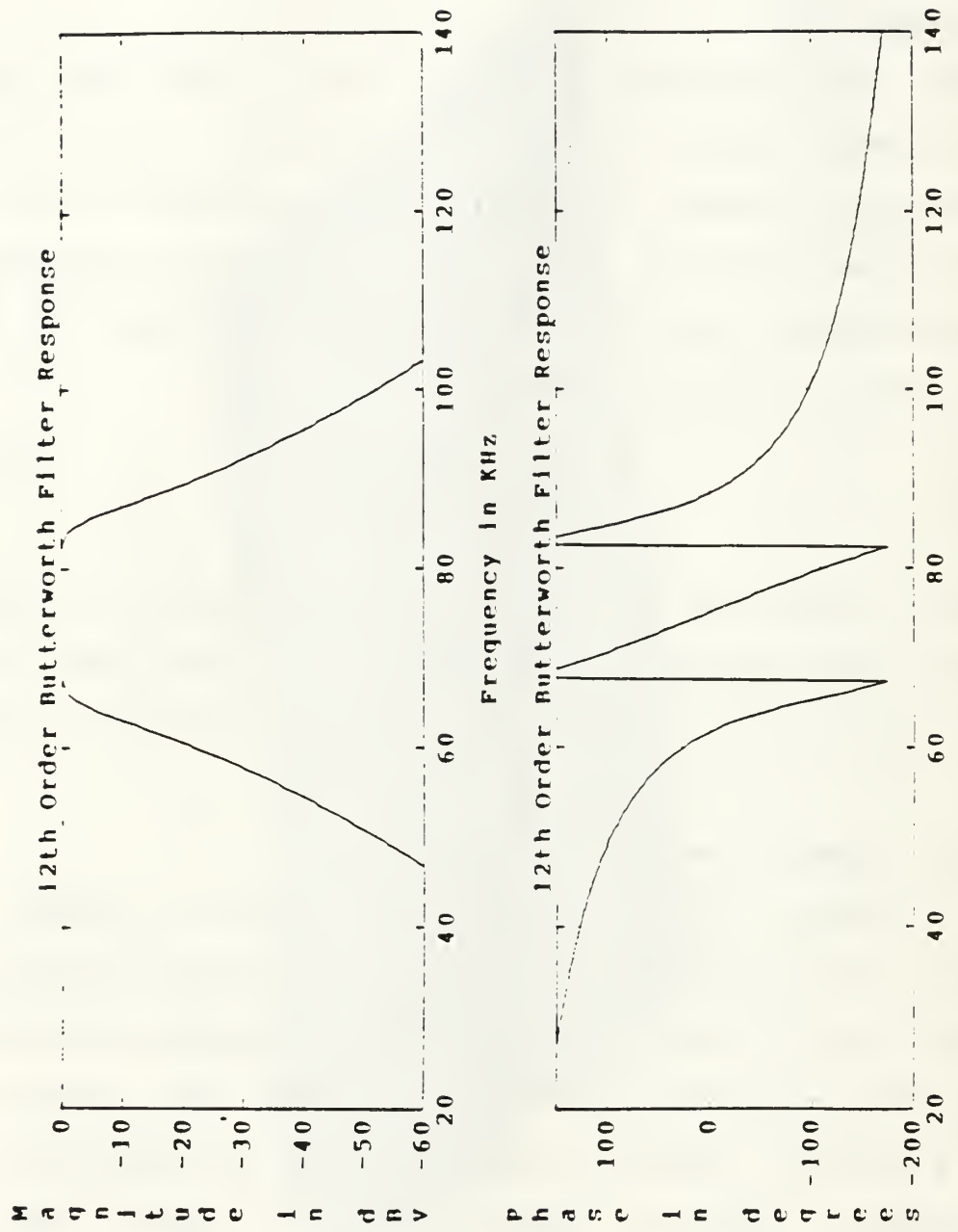


Figure 3. Magnitude and phase response of the hydrophone model.

Spherical spreading is a type of spreading loss in which the signal power is reduced as it propagates unbounded from a source in an unbounded medium. The signal is assumed to propagate radially, and the intensity of the signal decreases as the square of the range. As the power radiates outward it must cover increasingly larger areas; this reduces the power at any one point. The transmission loss due to spreading is

$$Loss = 10 \log r^2 = 20 \log r \qquad (2\text{-}2)$$

where r is the range from source to receiver. (Urick, 1983, pp. 100-101)

Unlike spreading loss, absorption loss varies logarithmically with range. At frequencies above 10 kHz, the attenuation of underwater signals due to sound absorption becomes significant. Figure 4 (Clay and Medwin, 1977) shows the attenuation profile of underwater acoustic signals in dB/m versus frequency in kHz. Thus, the absorption loss for a 75 kHz signal is

$$\alpha = 0.04 r \quad dB \qquad (2\text{-}3)$$

where r is the range.

Figure 5 depicts the power attenuation versus distance for a 75 kHz acoustic signal propagating through the test

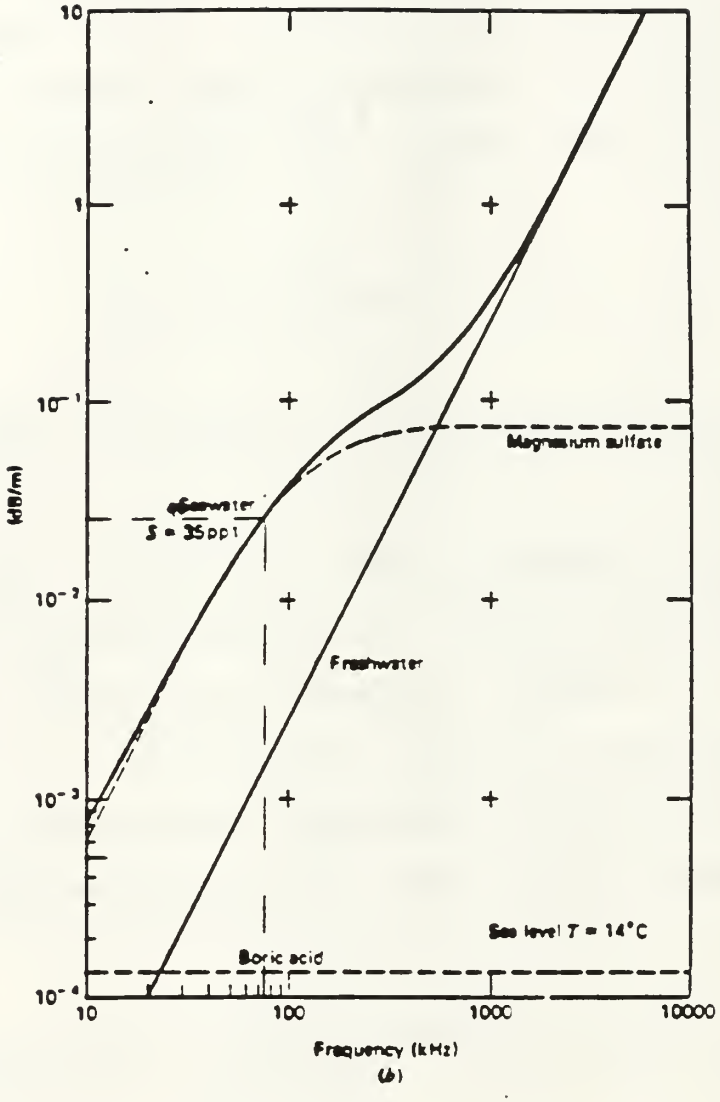


Figure 4. Attenuation profile. (Clay and Medwin, 1977, p. 101)

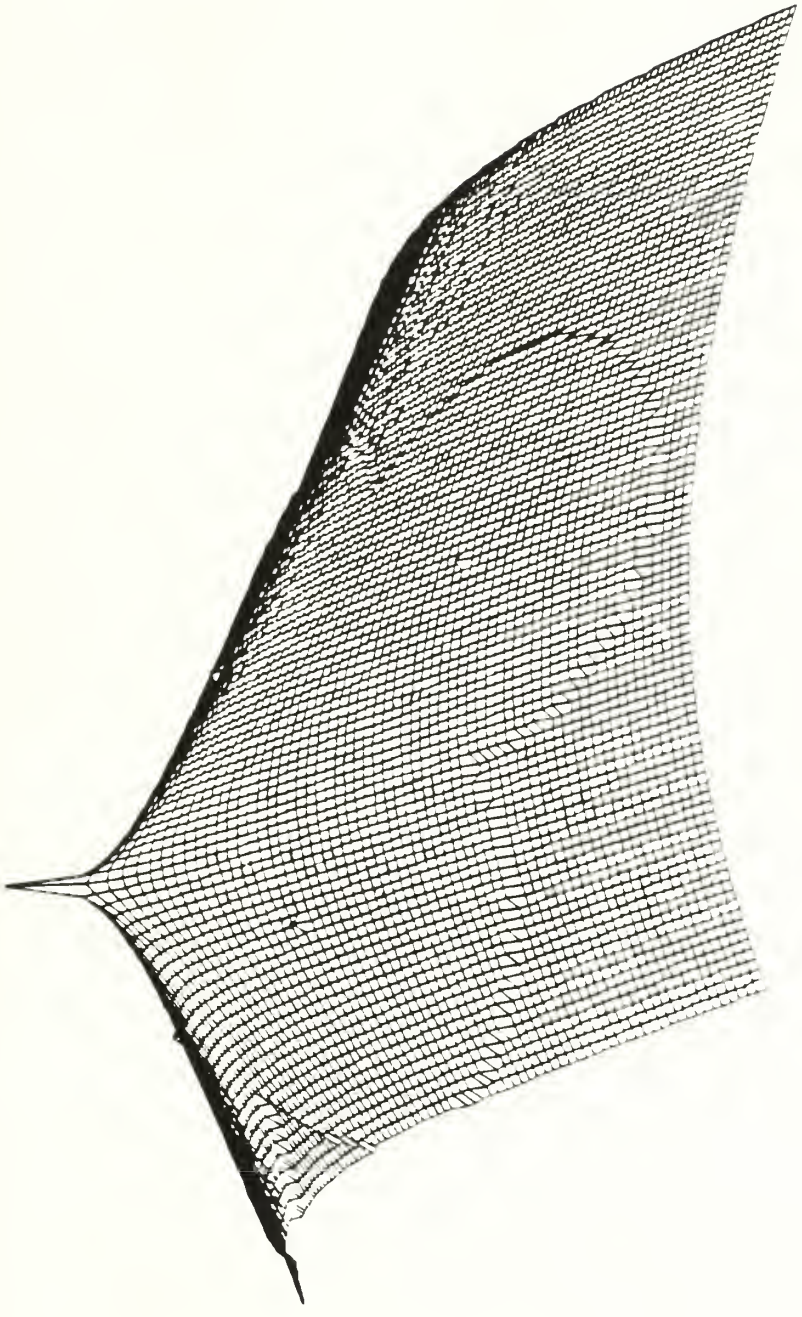


Figure 5. Mesh plot of the attenuated tracking and countermeasure signals.

range. Both types of losses, spherical spreading and attenuation, are accounted for in the illustration.

2. Propagation Delay

In salt water, sound travels at about 1500 meters per second (Urick, 1983, p.322). This relatively slow speed can cause significant delay times between transmission and reception of a signal. If a tracking signal source is at the outer edge of a hydrophone coverage area, the delay would be 1 s. Propagation delays of this order can destroy any correlation that might exist between signals received at two different hydrophones. Thus, since correlation between signals received at two different locations is a critical feature of adaptive filtering schemes, propagation delay becomes an important consideration. The delay time is given by

$$T = \frac{r}{1500} \qquad (2\text{-}4)$$

where T is in seconds and r is the range traveled in meters.

3. Multipath

Signals arriving at a receiver via multiple paths can cause constructive or destructive interference. Since the hydrophones are mounted on the ocean bottom, only the multiple paths caused by surface reflection of the signals is considered. These multipath effects are taken into account

when generating both the tracking and countermeasure noise signals. A delayed and attenuated version of the noise signal is added to the original noise signal. However, the tracking signal is transmitted at discrete time intervals and the reflected signal is typically received in an interval that does not overlap with the time interval in which the direct signal is received. Thus, after the reception of the direct path signal, the subsequently received reflected tracking signal can be identified and safely ignored.

4. Doppler Effect

The frequency of a received signal depends on the relative motion between source and receiver. The relationship between the transmitted frequency and the received frequency is given by (Urick, 1983, p. 322)

$$f' = f\left(\frac{c}{c+v}\right) \qquad (2\text{-}5)$$

where c is the sound speed and v is the relative speed of the vehicle. The frequency shift is

$$\Delta f \approx \pm 0.69 Hz/(knot)\text{-}(kHz). \qquad (2\text{-}6)$$

With a maximum speed of 50 knots (about 26 m/s) for the vehicle, the associated doppler shift at 75 kHz is ± 2.6 kHz. The sign of the shift depends on whether the signal source is moving toward or away from the hydrophone.

III. ADAPTIVE FILTERING

Adaptive filtering provides a method of recovering a desired signal in additive noise when the signal statistics are either unknown or are slowly varying with time. If the first and second order signal statistics were known, then a fixed optimal filter such as the Wiener filter could be used. However, without complete knowledge of the signal statistics, some form of estimation of those statistics is required. Two algorithms that help estimate filter coefficients under these conditions are least mean squares (LMS) and recursive least squares (RLS). (Haykin, 1984, p. 2)

The Wiener-Hopf algorithm provides the basis of these two methods. The Wiener-Hopf equation for estimating a finite impulse response filter weight vector $\underline{a}$ is given by (Haykin, 1984, p. 32)

$$\underline{a} = R_x^{-1} \underline{r}_{dx} \tag{3-1}$$

where R_x is the autocorrelation matrix of the input signal and $\underline{r}_{dx}$ is the cross-correlation vector between the desired signal d(n) and the received input signal x(n). Adaptive filtering schemes fundamentally try to approximate the weight vector of the optimal filter by adjusting it "on the fly".

A. LEAST MEAN SQUARES (LMS) ALGORITHM

The LMS algorithm is based upon the method of steepest descent and the knowledge that the mean-square error forms a paraboloid in the filter coefficient space. The error e(n) is defined as the difference between the desired signal d(n) and the filter output y(n) (see Figure 6)

$$e(n) = d(n) - y(n) \tag{3-2}$$

where

$$y(n) = \underline{x}^T(n)\underline{a} \tag{3-3}$$

and $\underline{x}(n)$ is the vector of observations (the received signal). Thus, the mean-squared error is

$$J = E[e^2(n)]. \tag{3-4}$$

A plot of J versus $\underline{a}$, called the error surface, is a paraboloid in shape. Expanding equation (3-4) gives

$$J = \sigma^2 - 2\underline{a}^T\underline{r}_{dx} + \underline{a}^T\underline{R}_x\underline{a}. \tag{3-5}$$

Differentiating the mean-squared error with respect to the weight vector produces the gradient vector ∇

$$\nabla = \frac{\partial J}{\partial \underline{a}} = -2\underline{r}_{dx} + 2\underline{R}_x\underline{a}. \tag{3-6}$$

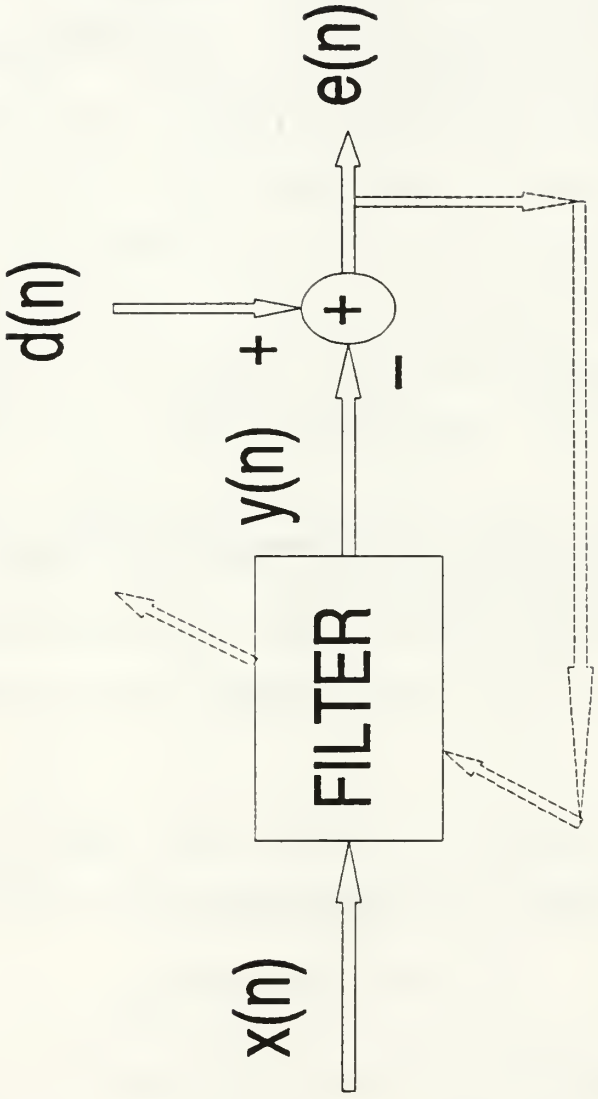


Figure 6. The basic adaptive filter model.

The gradient vector points in the direction of steepest ascent on the error surface. Based on the gradient search or steepest descent method the filter weight vector is recursively updated as follows

$$a(n+1) = a(n) + \frac{1}{2}\mu[-\nabla(n)] \qquad (3\text{-}7)$$

where - ∇(n) points the filter weight correction toward the minimum point on the error surface and μ is the step size parameter. Successive corrections in the direction of the negative gradient vector eventually minimize the mean-squared error and lead to an optimum filter weight vector. (Haykin, 1984, pp. 93-102)

B. RECURSIVE LEAST SQUARES (RLS) ALGORITHM

While the LMS algorithm is based on a stochastic formulation, the RLS algorithm is a finite data formulation based method. It is computationally more expensive than the LMS algorithm, but it is known to converge to the optimal solution significantly faster than the LMS algorithm. (Haykin, 1984, p. 149)

The sum of the squared errors between the desired signal d(n) and the filter output y(n) is given by

$$J = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (d_i - \underline{a}^T \underline{x}_i)^2 . \qquad (3\text{-}8)$$

Differentiating with respect to the weight vector and setting the resulting equation equal to zero gives

$$\frac{\partial J}{\partial a} = -2 \sum_{i} (d_i - \underline{a}^T \underline{x}_i) \underline{x}_i = 0 . \qquad (3\text{-}9)$$

Solving equation (3-9) leads to

$$R_n \underline{a}_n = \underline{r}_n \qquad (3\text{-}10)$$

where

$$R_n = \sum_{i=1}^{n} (\underline{x}_i \underline{x}_i^T) \qquad (3\text{-}10a)$$

is the input autocorrelation matrix and

$$\underline{r}_n = \sum_{i=1}^{n} (\underline{x}_i d_i) \qquad (3\text{-}10b)$$

is the cross-correlation vector between the filter input and the desired signal. Now, R_n and $\underline{r}_n$ can be written as

$$R_n = \sum_{i=1}^{n-1} (x_i x_i^T) + x_n x_n^T = R_{n-1} + x_n x_n^T \qquad (3\text{-}11)$$

and

$$r_n = \sum_{i=1}^{n-1} (x_i d_i) + x_n d_n = r_{n-1} + x_n d_n. \qquad (3\text{-}12)$$

Using equation (3-12) to rewrite equation (3-10) we have

$$R_n a_n = r_{n-1} + x_n d_n. \qquad (3\text{-}13a)$$

Substituting $R_{n-1}\underline{a}_{n-1} = \underline{r}_{n-1}$ and then adding and subtracting $\underline{x}_n\underline{x}^T{}_n\underline{a}_{n-1}$ on the right-hand side of (3-13a) yields

$$R_n a_n = R_n a_{n-1} + x_n\left(d_n - x_n^T a_{n-1}\right). \qquad (3\text{-}13b)$$

The term in parenthesis on the right-hand side of equation (3-13b) is a close approximation to the defined error and is denoted by $e_{(n|n-1)}$. Simplifying (3-13b) and solving for the weight vector gives

$$a_n = a_{n-1} + R_n^{-1} x_n e_{(n|n-1)}. \qquad (3\text{-}14)$$

Equation (3-14) provides a recursive formula with which to update the weight vector. However, evaluating R_n^{-1} is

computationally expensive. The matrix inversion lemma, commonly used to recursively update the inverse correlation matrix, is given by

$$R_n^{-1} = R_{n-1}^{-1} - \frac{R_{n-1}^{-1} \underline{x}(n) \underline{x}^T(n) R_{n-1}^{-1}}{1 + \underline{x}^T(n) R_{n-1}^{-1} \underline{x}(n)} . \qquad (3\text{-}15)$$

For notational simplicity, define

$$k_n \triangleq \frac{R_{n-1}^{-1} \underline{x}(n)}{1 + \underline{x}^T(n) R_{n-1}^{-1} \underline{x}(n)} = R_n^{-1} \underline{x}(n) . \qquad (3\text{-}16)$$

Substituting (3-16) into (3-14) and (3-15) yields

$$\underline{a}_n = \underline{a}_{n-1} + k_n \underline{e}_{(n|n-1)} \qquad (3\text{-}17)$$

and

$$R_n^{-1} = R_{n-1}^{-1} - k_n \underline{x}^T(n) R_{n-1}^{-1} \qquad (3\text{-}18)$$

respectively. Equations (3-17) and (3-18) provide recursive formulas for updating the weight vector. (Haykin, 1986, pp. 385-387)

C. RLS VERSUS LMS

There are some basic differences between the RLS and LMS algorithms. Both of these algorithms recursively adjust the weight vector to minimize the mean-squared error. However,

the algorithms differ in the way they update their respective weight vectors.

The LMS algorithm updates its weight vector based on the product of the error signal, the filter input, and the step-size parameter. Furthermore, the step-size parameter, along with the input signal, determines the speed of convergence for the weight vector. The choice of the parameter is constrained such that

$$0 < \mu < \frac{2}{\lambda_{max}} \qquad (3\text{-}19)$$

where λ_{max} is the largest eigenvalue in the correlation matrix R_x. The product of the scalar step-size parameter and the gradient vector produces a vector with which the filter weight vector is adjusted. (Haykin, 1984, pp. 100-103)

While the LMS filter weights are adjusted using a scalar transformation of the observations, the RLS filter weights are adjusted using a matrix transformation. Also, the LMS adjustment vector contains information available from the current iteration only, while the matrix used in adjusting the RLS filter weights is an estimate of the autocorrelation matrix for all of the past data. Therefore, the RLS filter weights undergo a higher order adjustment then the LMS filter weights, causing the RLS method to converge faster. (Haykin, 1984, pp. 148-149).

The superior performance of the RLS algorithm, however, is attained at the expense of increased computational complexity. The number of multiplications per iteration required for the RLS algorithm is on the order of $3M(3 + M)/2$ where M is the filter order. In contrast, the multiplications per iteration required for the LMS algorithm is on the order of $2M + 1$. (Haykin, 1984, p. 149)

IV. ADAPTIVE FILTERS FOR THE NUWES TEST RANGES

This chapter discusses two adaptive filtering schemes, the adaptive noise canceler and the adaptive line enhancer. Each scheme is developed from a simple lossless model to a complete model for the NUWES test range. Simulation results are presented for various scenarios in this development. Note that the term "noise" used throughout the chapter refers to the countermeasure signal, and the signal-to-noise ratio (SNR) is defined as

$$SNR = 10\,LOG_{10}\left[\frac{E[x^2(n)]}{\sigma^2}\right] \qquad (4\text{-}1)$$

where $E[x^2(n)]$ is signal power, and σ^2 is the noise (countermeasure) power.

A. ADAPTIVE NOISE CANCELER

The adaptive noise canceler consists of two inputs. The first input, the primary input, contains the desired signal corrupted by noise. The second input, or reference input, is derived from a source at which the tracking signal is weak, but contains noise which is highly correlated with the noise in the primary input. The reference signal is adaptively filtered to maximize its correlation with the primary signal,

then subtracted from the primary input to cancel the countermeasure noise. (Widrow, 1975, 1692-1693)

1. Noise Cancellation of an Ideal BPSK Signal

The adaptive noise canceler configuration for a 75 kHz BPSK signal in broadband white noise has the form depicted in Figure 7. Since the signal and noise have comparable propagation delay times, zero propagation delay is assumed for both the signal and countermeasure noise in the initial simulation. The signal received at the tracking array is the tracking signal corrupted by countermeasure noise. At the same time, the reference array receives only the countermeasure noise. Once received, the signals are passed to the adaptive noise canceler as the primary and reference signals, respectively. Figure 8 displays the time and frequency plots of the received and recovered signals. As seen in the signal spectrum plots, the noise that was present in the input signal is attenuated in the recovered signal.

2. Doppler Shift with ANC

The ANC scheme shown in Figure 7 can be used to process the doppler shifted BPSK signals. The noise canceler does not remove the doppler frequency shift, but removes the noise from the frequency shifted tracking signal. Figure 9 illustrates the recovery of a doppler shifted BPSK signal.

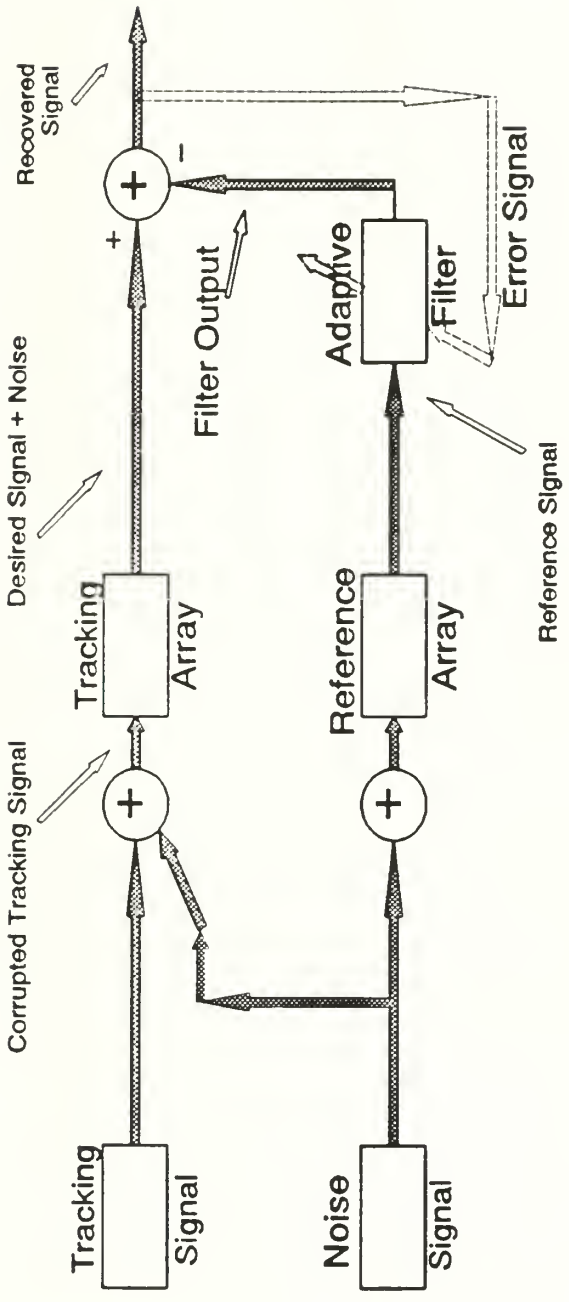


Figure 7. Adaptive noise cancellation scheme for the basic BPSK signal.

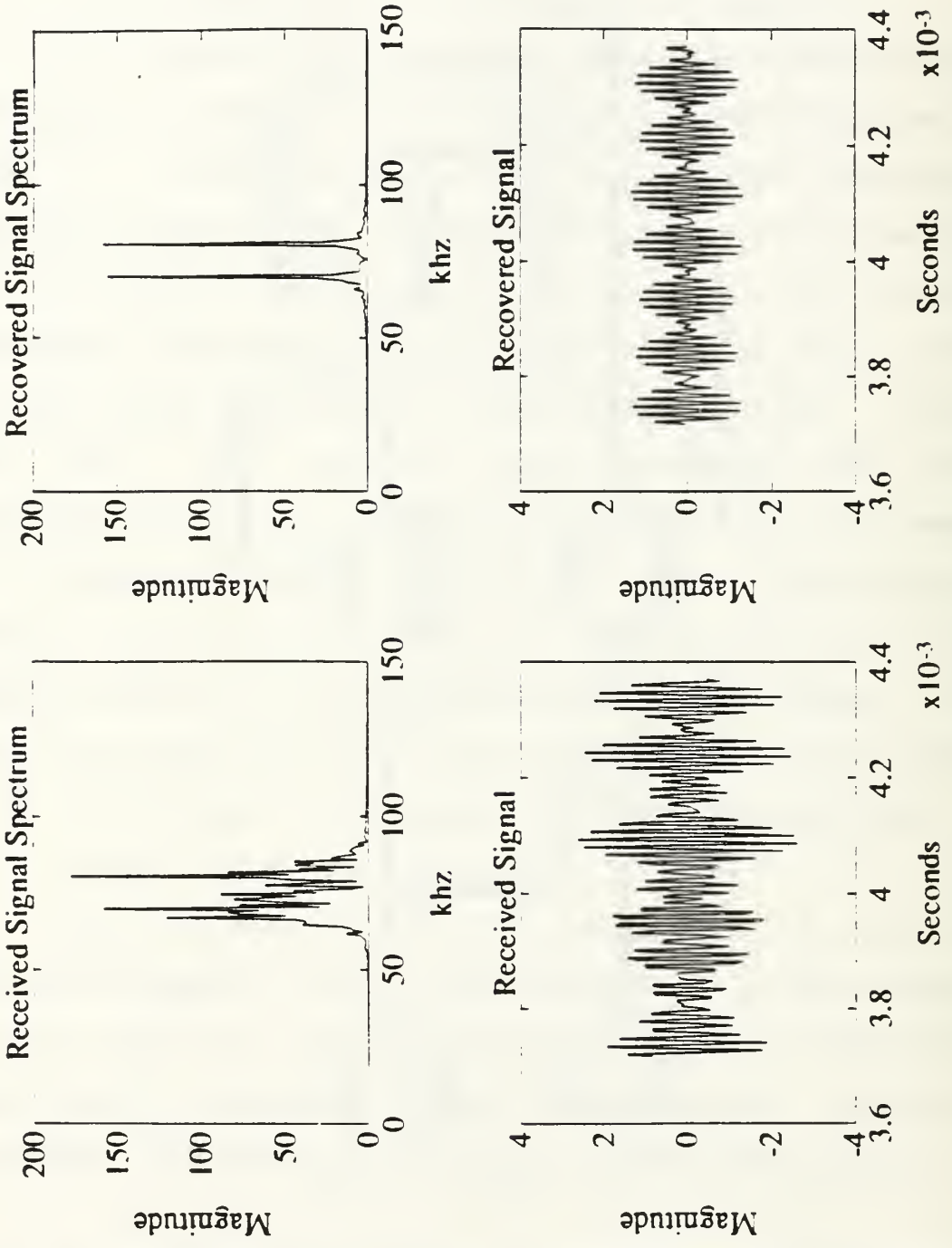


Figure 8. Received and recovered signals for the basic noise cancellation scheme (SNR = -10 dB).

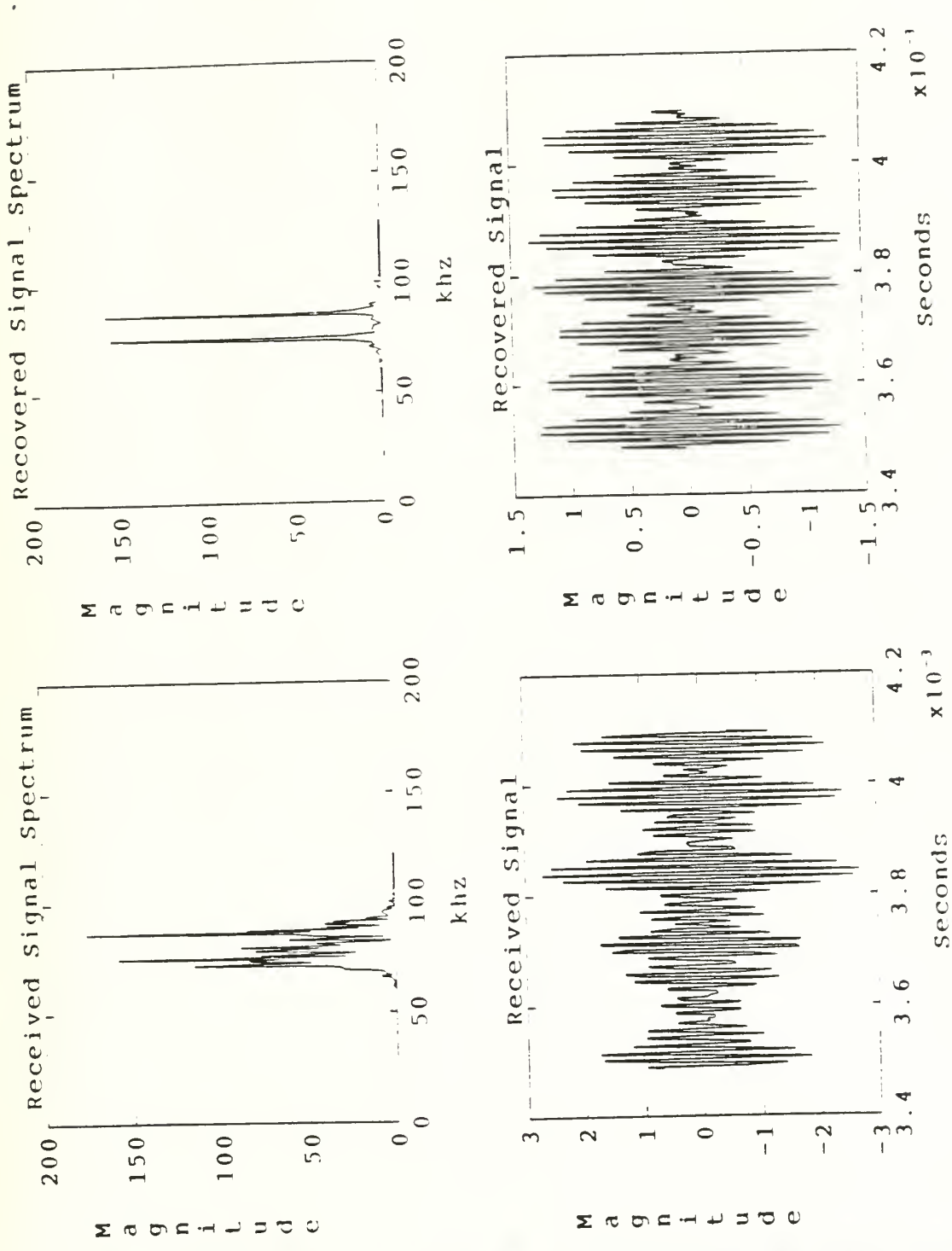


Figure 9. Received and recovered signals for the basic noise canceler with a doppler shifted signal (SNR = -10 dB).

3. Multipath Effects on ANC

Signals transmitted from an underwater source can propagate to the receiver via multiple paths. In this thesis, it is assumed that the surface reflected signal and the direct path signal comprise the multipath signal. The reflected signal is a delayed, sign inverted, and attenuated version of the direct path signal. Both the countermeasure noise signal and the tracking signal travel via these multiple paths. The reflected countermeasure noise increases the overall noise variance, the amount of which depends on the path lengths and the associated attenuation.

Unlike the countermeasure noise signal, the tracking signal is sent in discrete bursts with quiet time in between bursts. As such, the reflected tracking signal, when received, does not overlap in time with the direct path signal and can be disregarded by the NUWES tracking station. Figure 10 depicts a noise cancellation scheme incorporating the multipath situation. Since the noise is white, the multipath noise in the desired signal is uncorrelated with noise in the reference. The received signal spectrum in Figure 11 illustrates the addition of correlated and uncorrelated noise to the desired signal. In the recovered signal spectrum, the noise uncorrelated with the reference is still present.

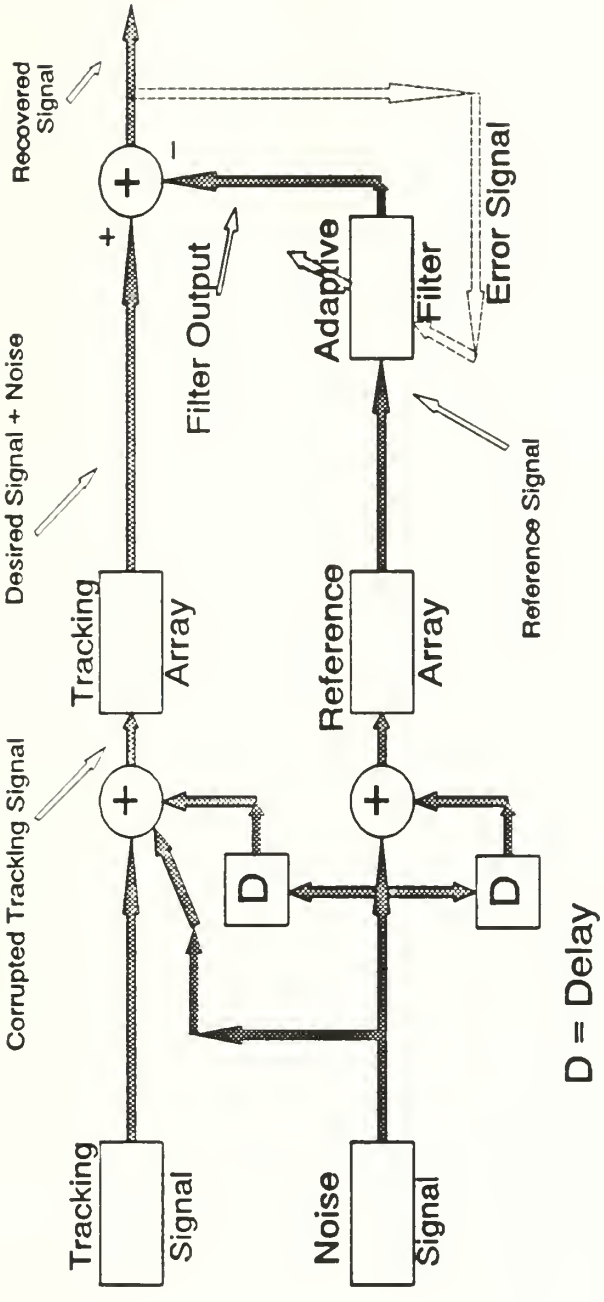


Figure 10. Noise cancellation scheme incorporating multipath.

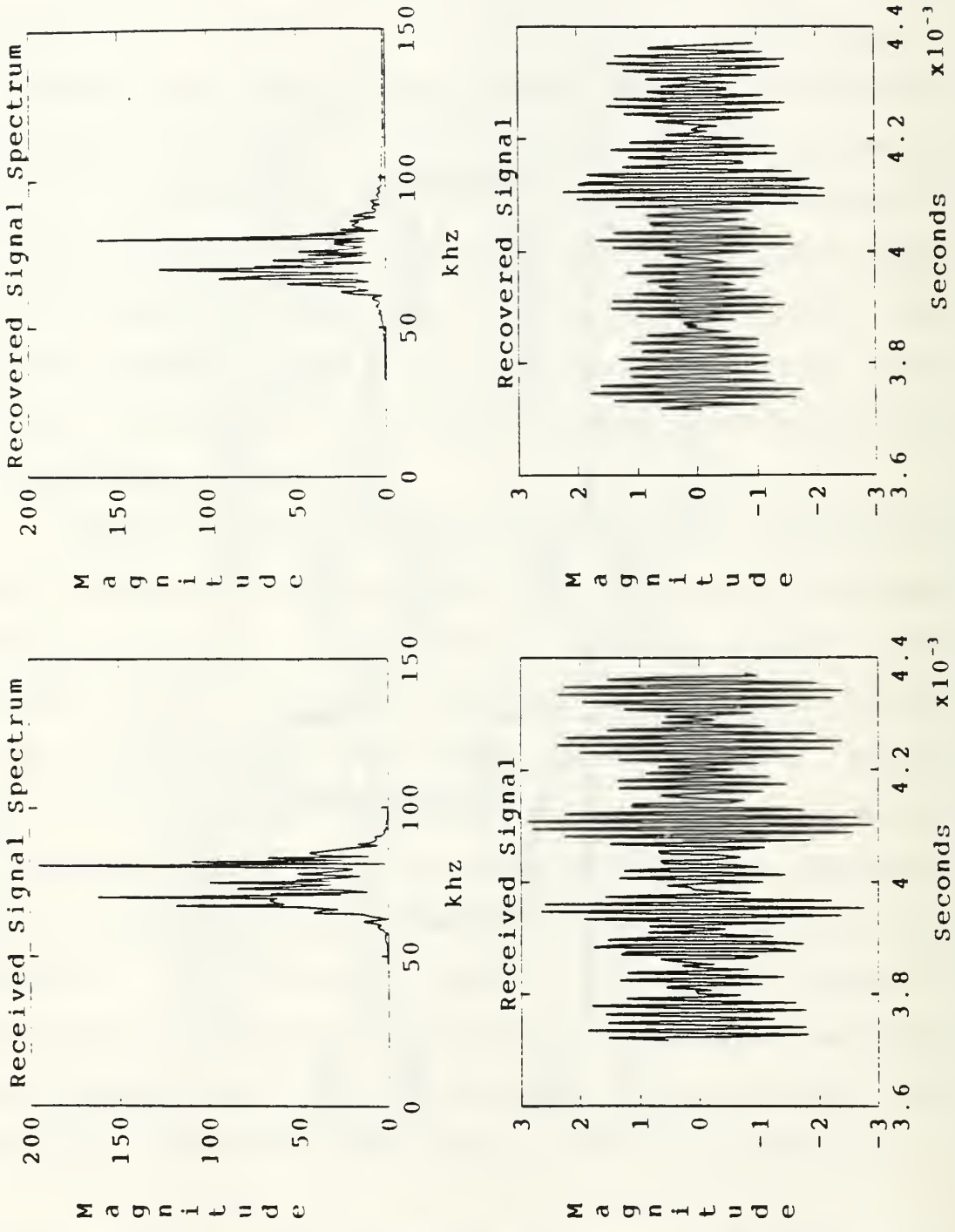


Figure 11. Received and recovered multipath signals (SNR= -10 dB).

4. Signal Propagation Delay

The ANC functions on the premise that some correlation exists between the noise in the primary signal and the noise in the reference signal. However, signal propagation delay destroys the correlation between broadband noise signals. Therefore, to allow the noise canceler to function with delay, the noise correlation must be restored. To reestablish the noise correlation, either the primary or the reference signal must be appropriately delayed. For realtime tracking, the delay must be applied to the reference signal. Delaying the primary signal would not permit realtime processing. Therefore, if delaying the reference noise, the countermeasure noise propagation time to the reference receiver must be shorter than the propagation time to the primary receiver. Thus, some form of local reference is required for the countermeasure noise. The local reference can be obtained by placing a receiver near the countermeasure noise source.

If realtime tracking were not a consideration then one of the nearby non-tracking hydrophone subarrays with a large noise signal could be used as the reference receiver. In this case, the first signal received, primary or reference, would be delayed to achieve maximum correlation.

5. The Overall Model for ANC

Figure 12 illustrates the overall simulation scheme incorporating delays, multipath, and correlation. The

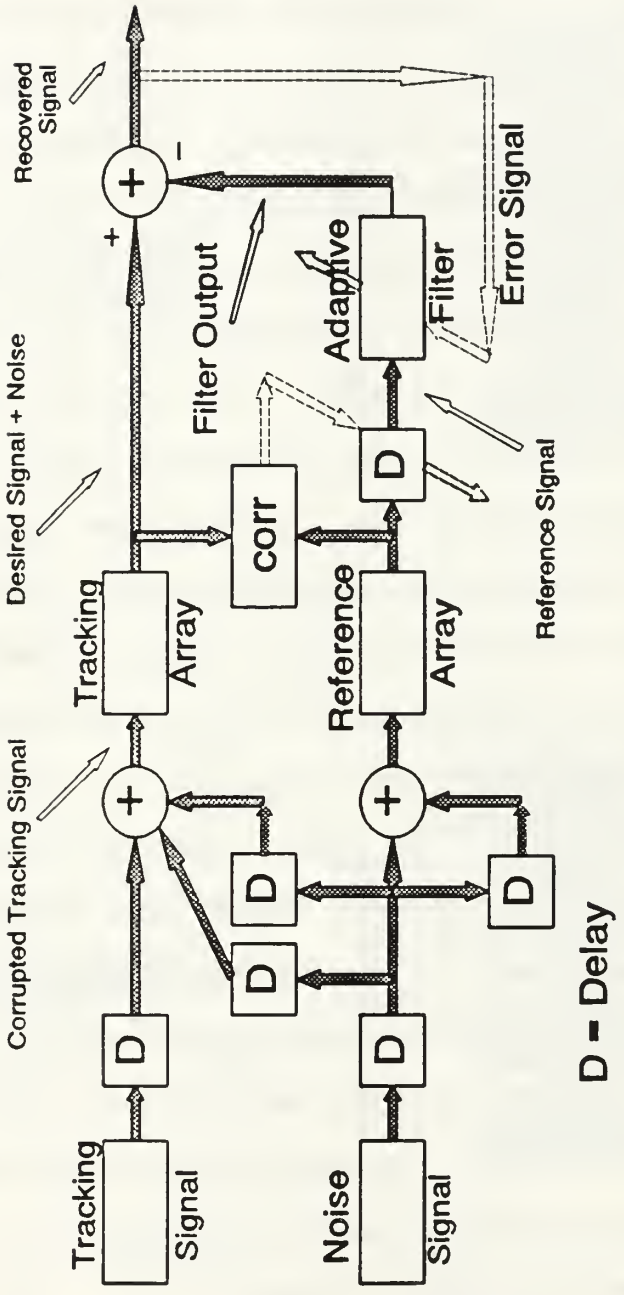


Figure 12. The complete noise cancellation scheme.

tracking signal and the countermeasure noise propagate with delay to the tracking array. In addition, a reflected, attenuated, delayed version of the countermeasure is received at the tracking array. The sum of these three signals comprise the corrupted tracking signal, which is in turn passed through a bandpass filter representing the tracking array. This filtered signal represents the primary input to the adaptive filter. The reference signal varies from the primary signal because it contains no tracking signal.

The correlator depicted in Figure 12 adjusts the delay of the signal received at the reference array to maximize the correlation between the primary input signal and the reference input. The error between the filter output and the primary input signal becomes the recovered tracking signal.

Figure 13 shows the output of the system at a signal-to-noise ratio (SNR) of -10 dB. (Additional graphs at different values of SNR are included in Appendix A.) This scenario presumes that correlation between the noise in the primary signal and the reference signal exists only between the direct path noise. Therefore, the recovered tracking signal contains the uncorrelated, surface-reflected noise.

B. ADAPTIVE LINE ENHANCER

The Adaptive Line Enhancer (ALE), illustrated in Figure 14, differs from the adaptive noise canceler in that the reference signal x(n) is derived from the input signal d(n)

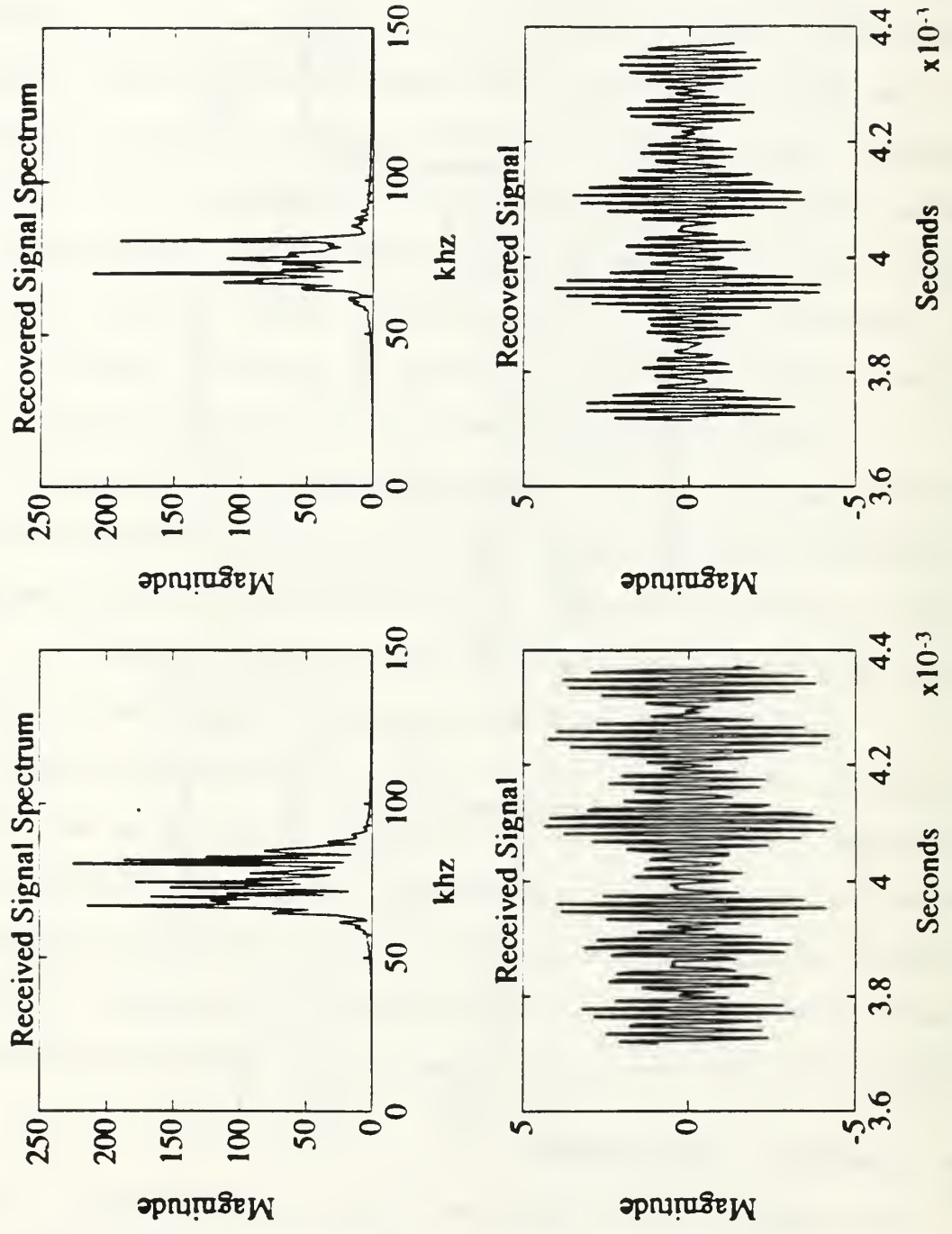


Figure 13. The received and recovered signals for the complete noise cancellation scheme (SNR = -10 dB).

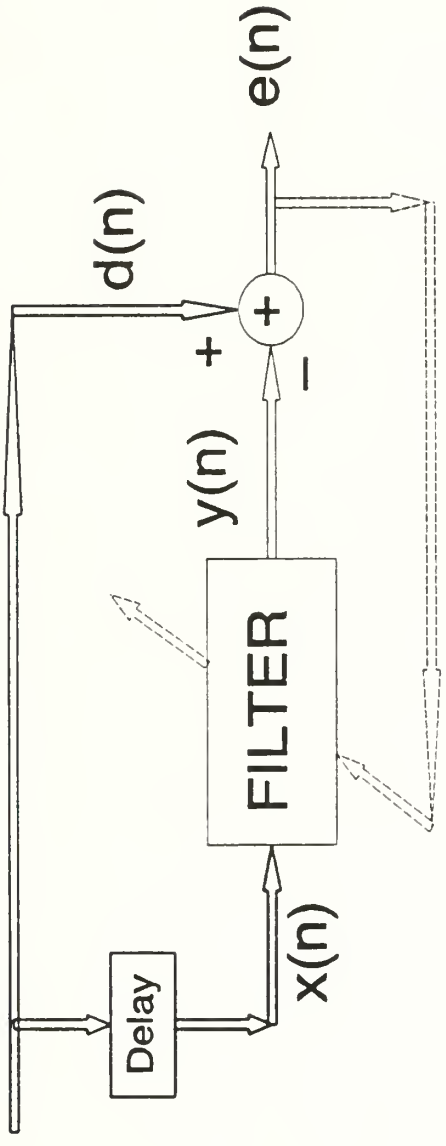


Figure 14. The basic adaptive line enhancer.

using a delay.

The input signal is a narrowband signal corrupted by white noise. Since white noise has a narrow autocorrelation function (theoretically an impulse), the delay used in producing the reference signal removes the correlation between the noise components of the primary and reference signals. However, the narrowband signal has a wider autocorrelation function and retains its correlation despite the delay.

The output of the adaptive filter is the correlated portion of the primary and reference signals. This output is subtracted from the primary input signal to minimize the error signal.

The ALE scheme used to recover the tracking signal is shown in Figure 15. The input to the ALE is the sum of the BPSK tracking signal and the broadband countermeasure noise. However, the filtering effect of the hydrophone reduces the bandwidth of the countermeasure noise to about 20 kHz. Therefore the noise in not broadband, nor is the tracking narrowband. In fact, the noise bandwidth is equal to the bandwidth of the tracking signal. Since the ALE is effective only with a narrowband signal in broadband noise, the ALE will not be able to recover the tracking signal. Figure 16 shows the output of the ALE when trying to recover the tracking signal.

A whitening filter inserted after the hydrophone will not improve the recovery of the tracking signal. Since after the

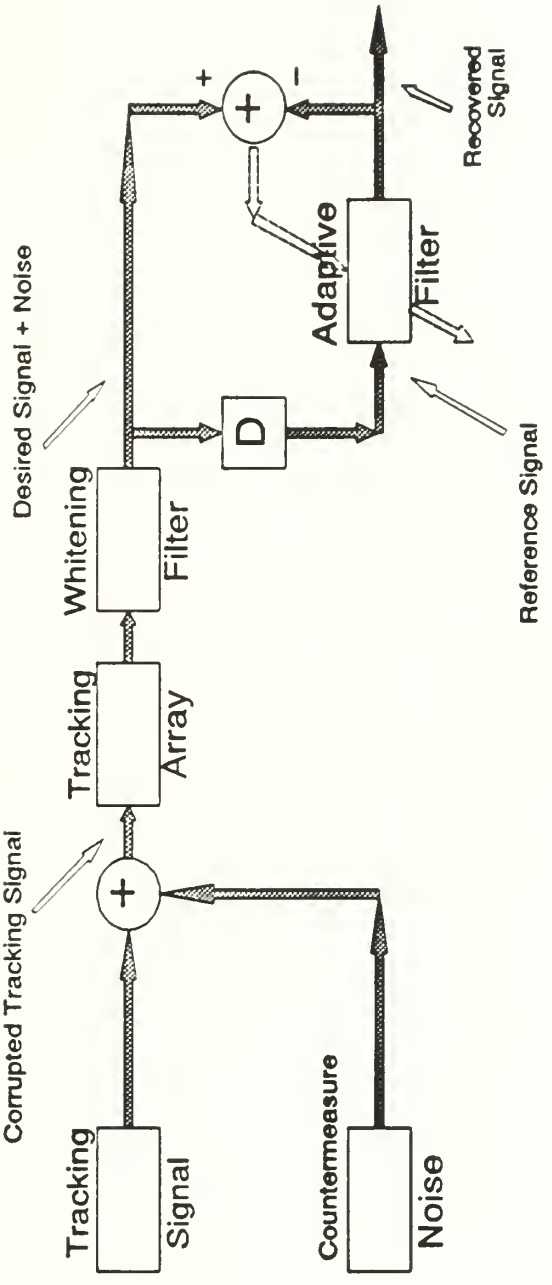


Figure 15. The ALE scheme for recovering the basic signal.

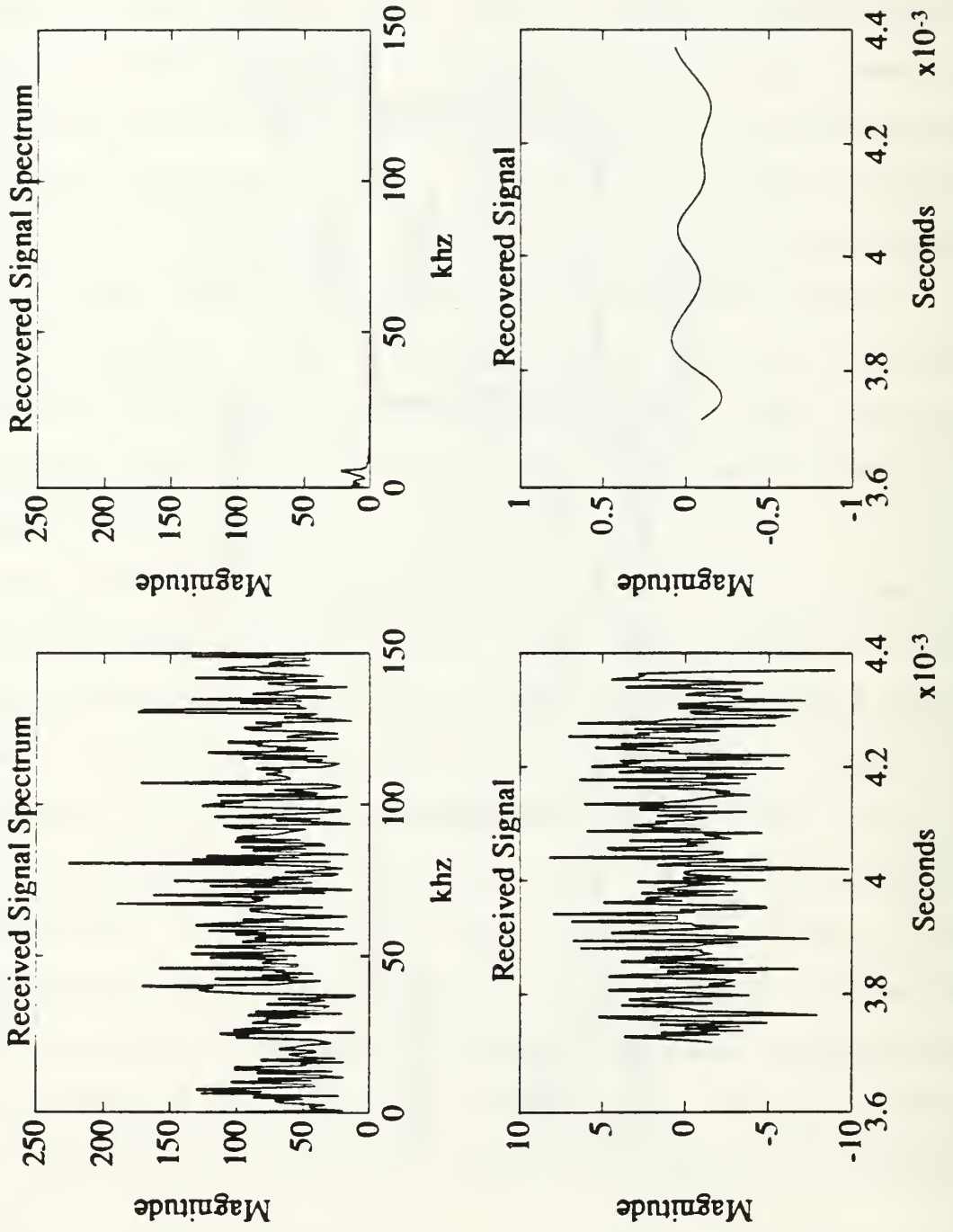


Figure 16. The received and recovered signals for the basic adaptive line enhancer scheme (SNR = -10 dB).

hydrophone, both signals have the same bandwidth, the whitening filter broadens both signals, and the ALE will not be able to recover the tracking signal.

C. PROBABILITY OF ERROR COMPARISONS

Monte Carlo simulations were run using the complete signal models developed above. The simulations used signal-to-noise ratios of 0, -5, -10, -15, and -20 dB. The recovered signals were heterodyned with a coherent source then low pass filtered to remove the higher frequency components. Once the bits were recovered, they were compared with the original bit pattern to determine the number of bits in error.

At each SNR level, for both ANC and ALE schemes, the process was repeated 100 times. The tracking signals were represented by a random bit pattern of 47 bits. Thus, at each SNR, there were 4700 bits. The total number of bits in error throughout the 100 trials were accumulated. Figures 17 and 18 display the percentage of bits correctly recovered as a function of SNR. On both figures, the 'x's mark individual recovery percentages for each of the 100 trials while the solid line is the overall percentage of bits correctly recovered. Also, for comparison, probability of error results were obtained for the case in which no adaptive filter was used. From this case, the ALE and the ANC schemes are compared as shown in Figure 19.

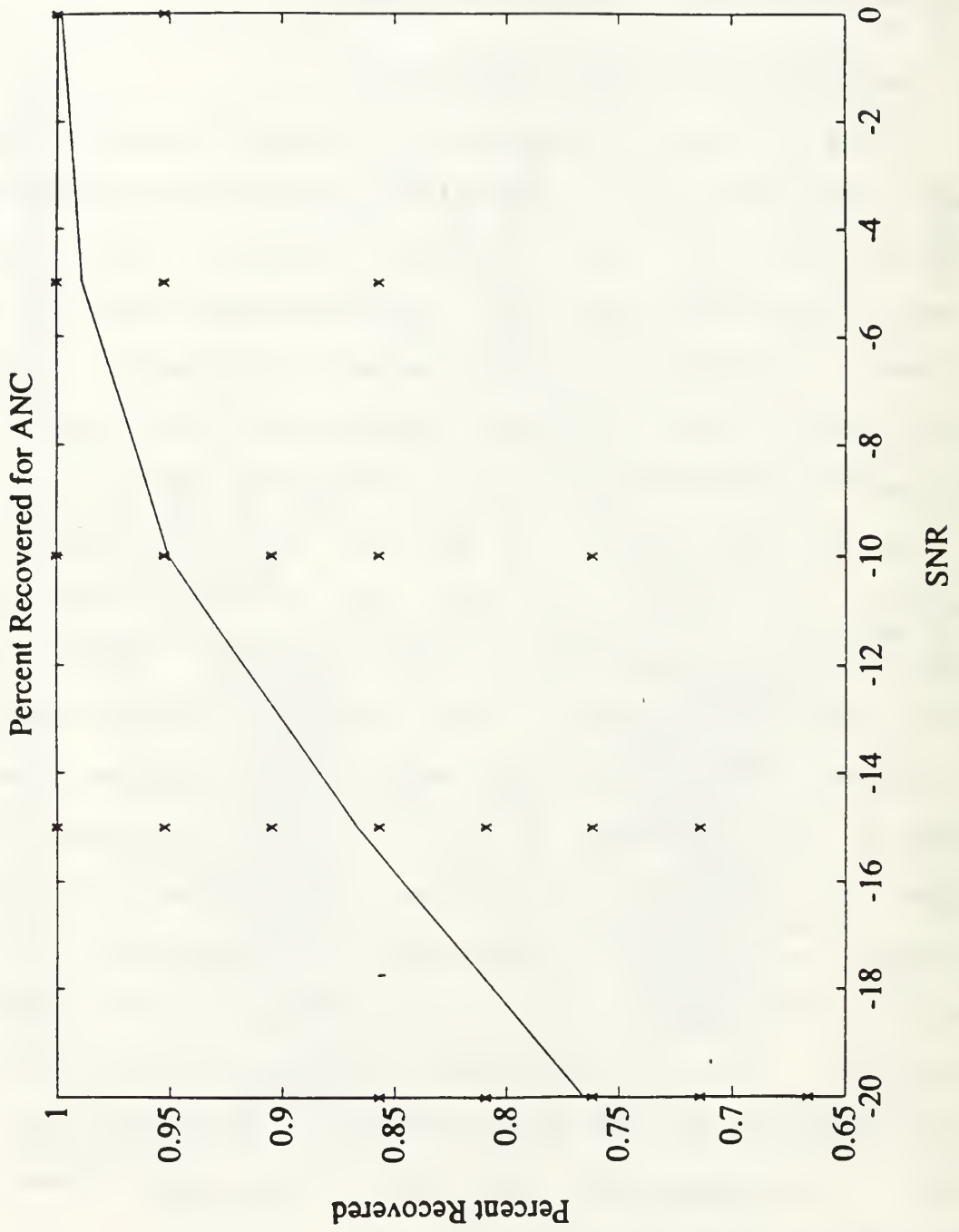


Figure 17. Probability of error for the adaptive noise cancellation scheme.

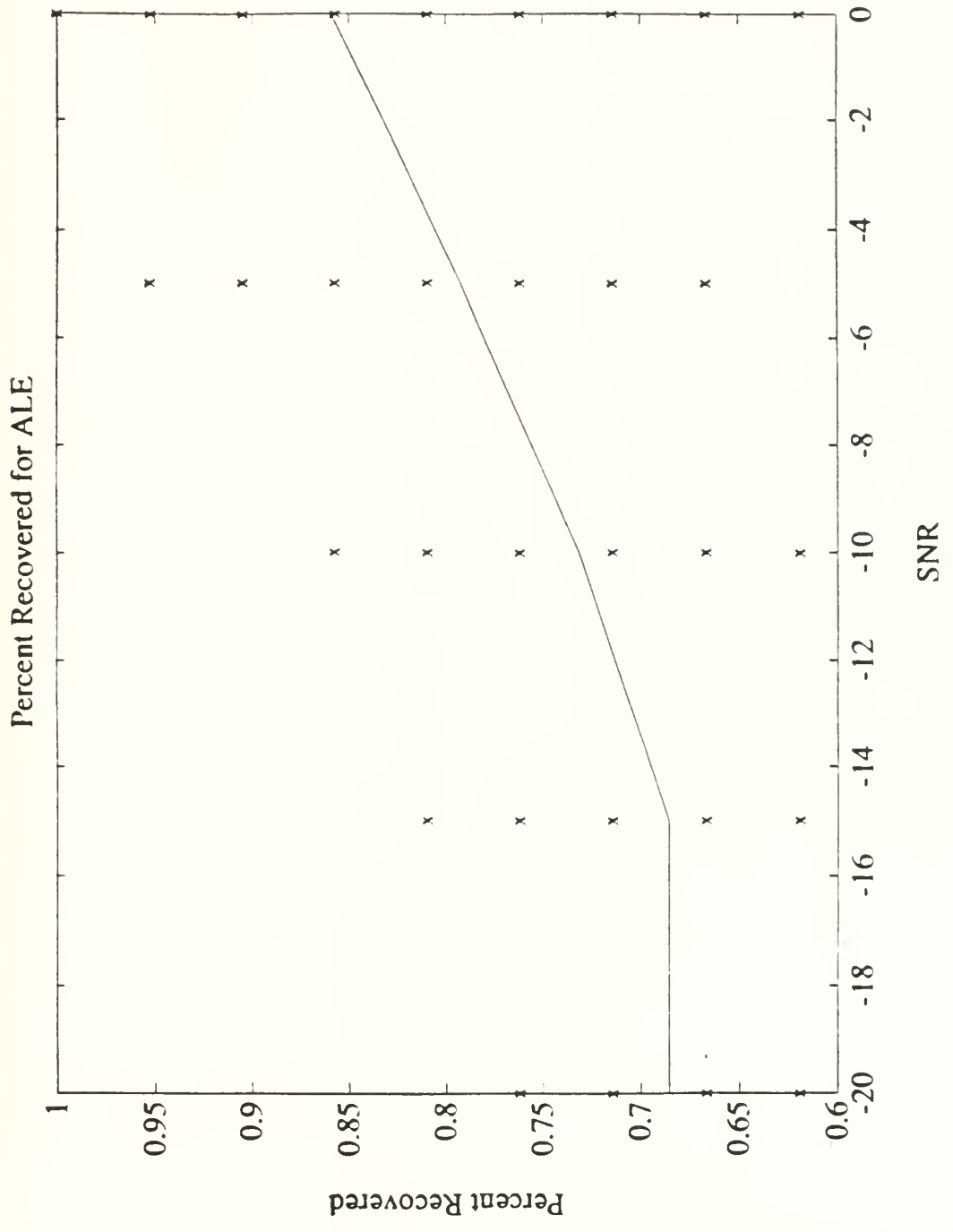


Figure 18. Probability of error for the adaptive line enhancer scheme.

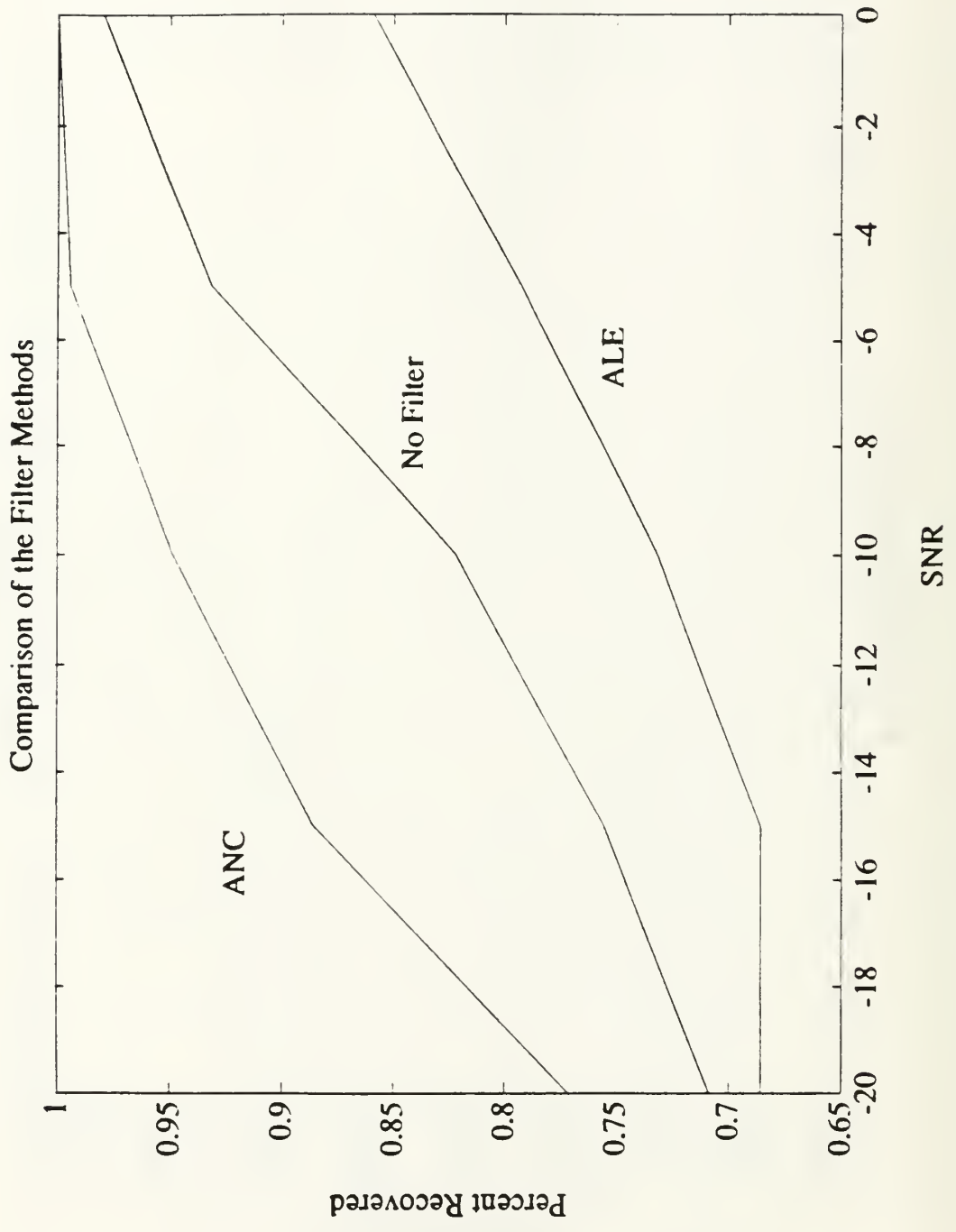


Figure 19. Probability of error comparison for the filtering schemes versus the 'no filter' case.

The adaptive noise canceler consistently provided about 4 dB of signal gain above the 'no filter' case. At 0 dB SNR the probability of an error is about zero percent. As the SNR is decreased to -20 dB, the probability of error increased to about 23 percent.

On the other hand, the adaptive line enhancer attenuated rather than recovered the tracking signal. The performance of the ALE for this type of signal was worse than if no filter had been used. This poor performance was due to the broadband nature of the tracking signal.

V. CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY OF RESULTS

This thesis investigated adaptive filtering for the recovery of vehicle tracking signals in the presence of countermeasure noise. A model of the NUWES test ranges was developed to which the adaptive filtering schemes were applied. This model included a 75 kHz BPSK signal to simulate the tracking signal and white noise to represent the countermeasure. These signals were subjected to propagation losses, propagation delays, multipath effects, and doppler shifts.

Two adaptive filtering methods were examined: adaptive noise cancellation and adaptive line enhancing. The first scheme, adaptive noise cancellation, employs two inputs. The first input, the primary input, contains the tracking signal corrupted with countermeasure noise. The second input, the reference input, contains countermeasure noise only. Since the noise in the reference input is correlated with the noise in the primary input, the ANC cancels the noise, thereby improving the received tracking signal.

In the simulation, a reference receiver was placed near the countermeasure noise source to ensure that the propagation time for the reference input was shorter than that for the

primary input. Therefore, the delay used in aligning the signals was applied to the reference and not to the primary input, allowing realtime processing of the tracking signal.

Results of a Monte Carlo simulation are depicted in Figure 19. The horizontal distance between the ANC line and the 'no filter' line is a measure of the processing gain. The adaptive noise canceler provided about 4 dB of processing gain across the 0 to -20 dB SNR range. The results show that the correlator used in the noise canceler was unable to correlate the surface reflection signals.

The second filtering scheme investigated, the adaptive line enhancer, recovers narrowband signals from corrupting broadband noise. However, the BPSK tracking signal used by the NUWES test stations has a bandwidth of over 20 kHz. Furthermore, the filtering effect of the hydrophone diminishes the bandwidth of the countermeasure noise to 20 kHz. The similarity of these two bandwidths prevents the ALE from recovering the tracking signal. Figure 19 shows that the line enhancer fails to provide any signal gain. In fact, signal recovery worsened when using the line enhancer.

B. RECOMMENDATIONS

For the model developed in this thesis, the adaptive noise canceler provides significant noise attenuation. However, to implement the filter, minor modifications to existing range equipment must be made. To track in realtime, some form of

local countermeasure noise receiver must be used to furnish the reference input. If realtime tracking were not a consideration, then one of the nearby non-tracking hydrophone subarrays with a large noise signal could be used as the reference receiver. In this case, delay is applied to whichever signal is received first, primary or reference, to achieve maximum correlation.

The adaptive line enhancer would be easier to implement at the NUWES test ranges as it requires only one receiving hydrophone array. However, an adaptive line enhancer is effective only with a narrowband signal in wideband noise (Haykin, 1984, pp. 18-19, Widrow, 1975, pp. 1711-1712). Thus, recovery of the signal in its present form is not feasible with the line enhancer. However, if the signal could be spectrally compressed into a narrowband form, adaptive line enhancement could be a viable filtering scheme.

APPENDIX A. ADDITIONAL RESULTS FOR THE ANC

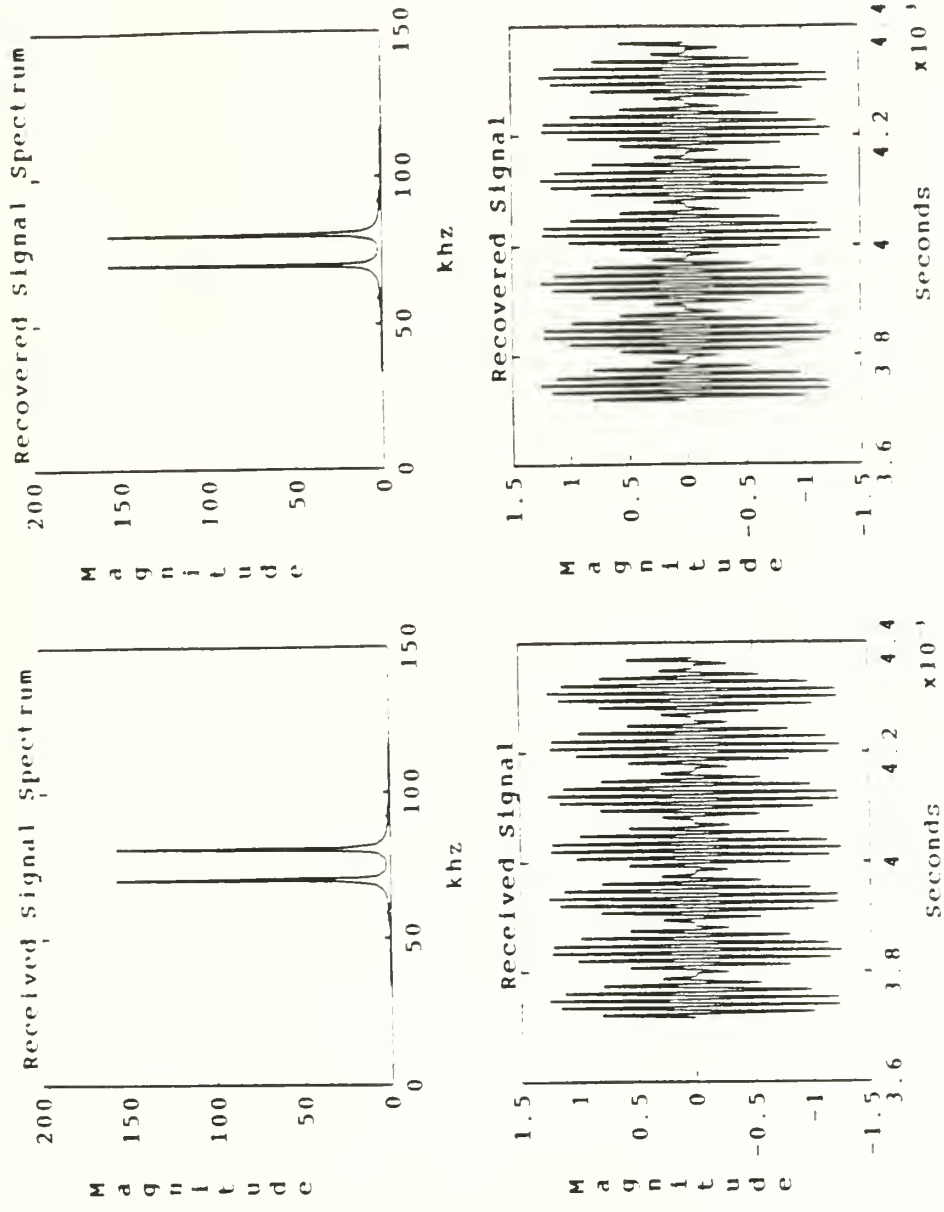


Figure 20. The received and recovered signals with no noise.

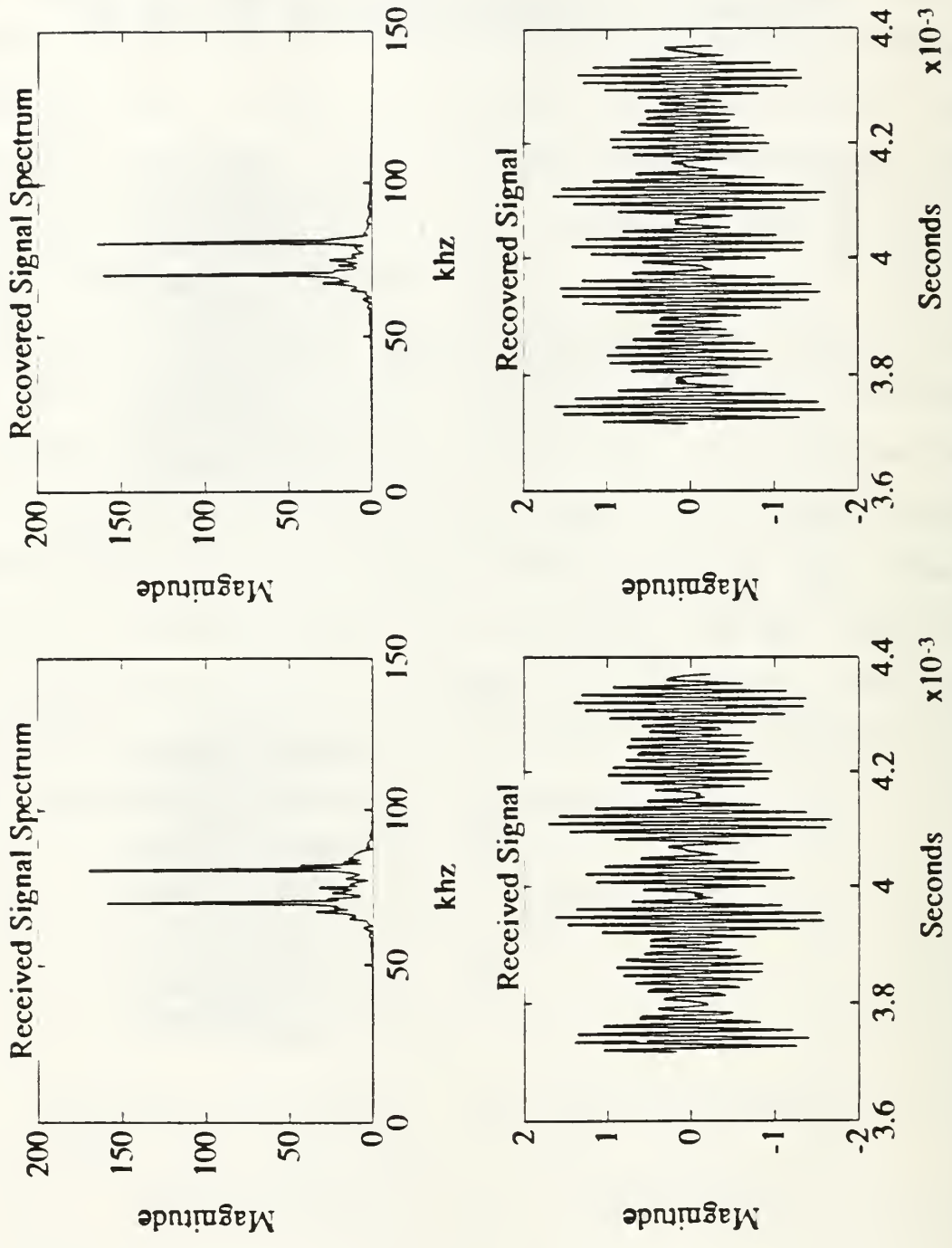


Figure 21. The received and recovered signals for an input SNR of 0 dB.

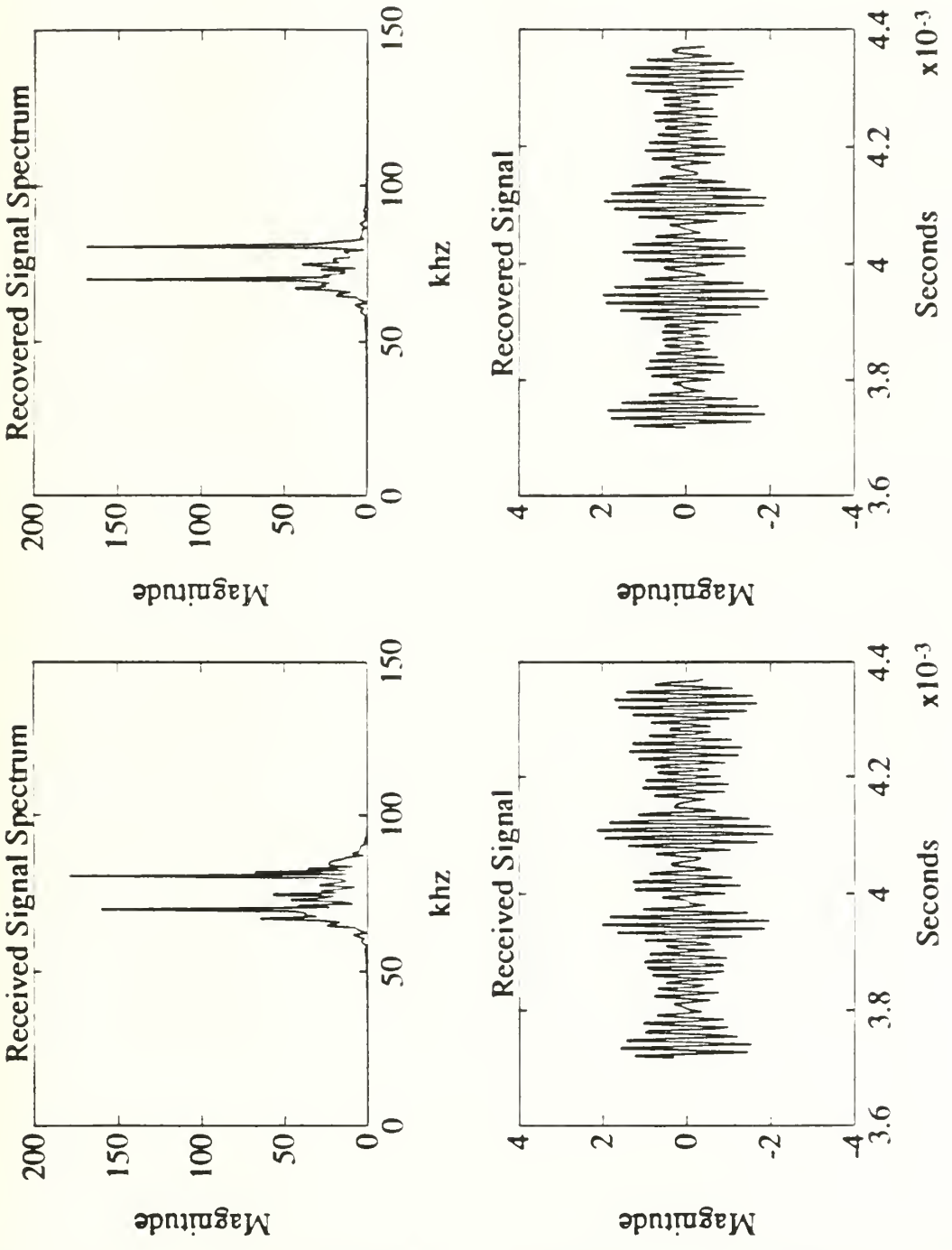


Figure 22. The received and recovered signals for an input SNR of -5 dB.

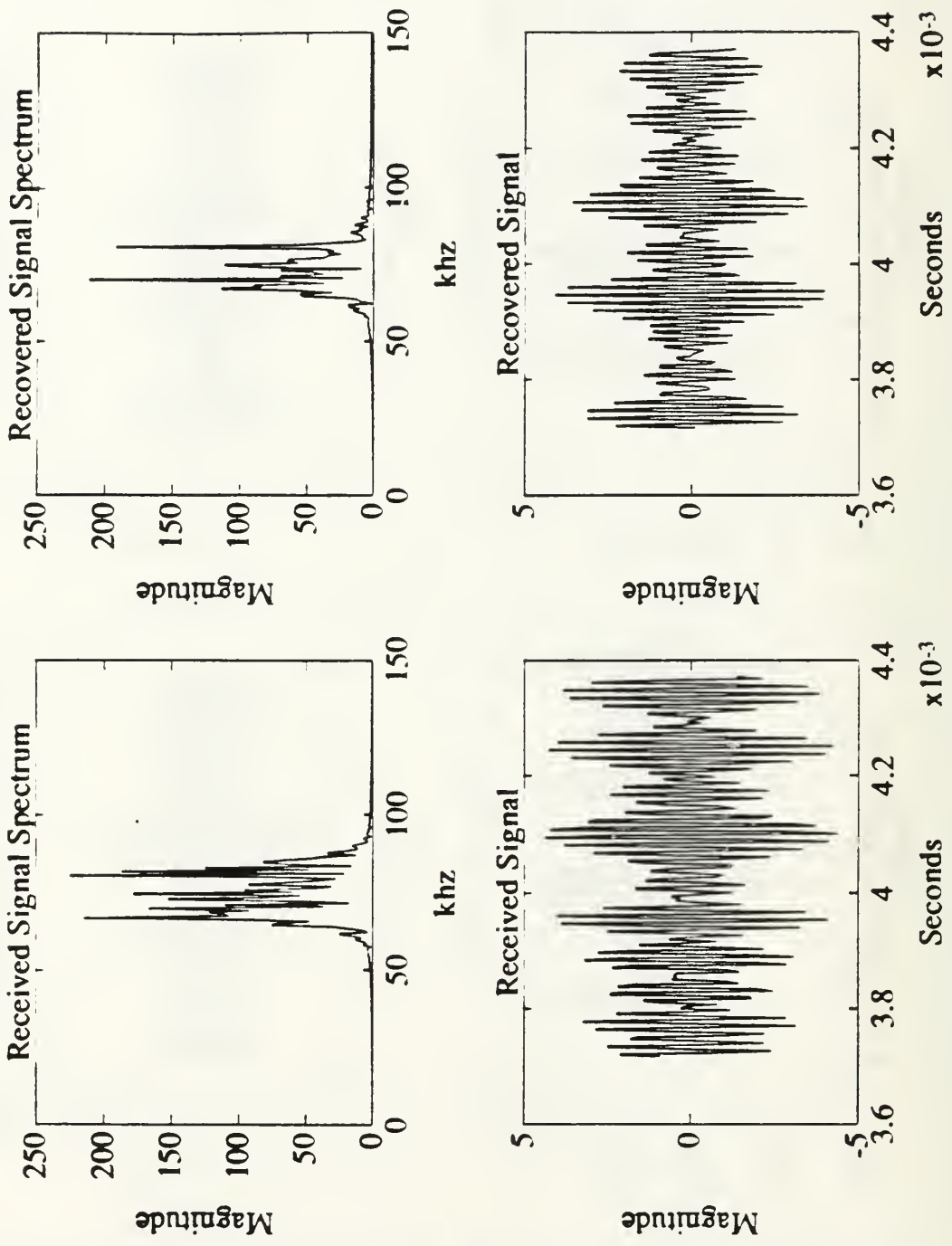


Figure 23. The received and recovered signals for an input SNR of -10 dB.

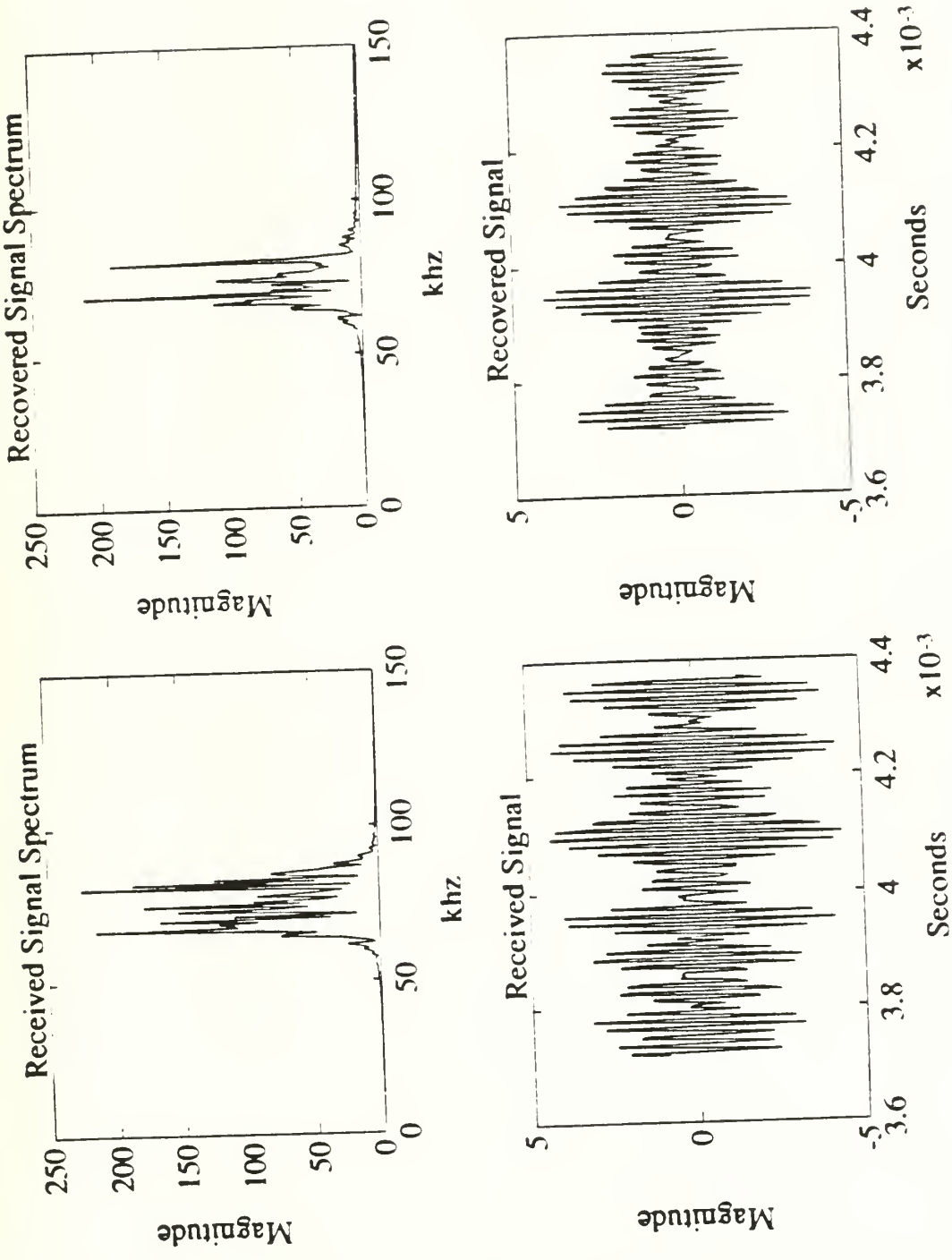


Figure 24. The received and recovered signals for an input SNR of -15 dB.

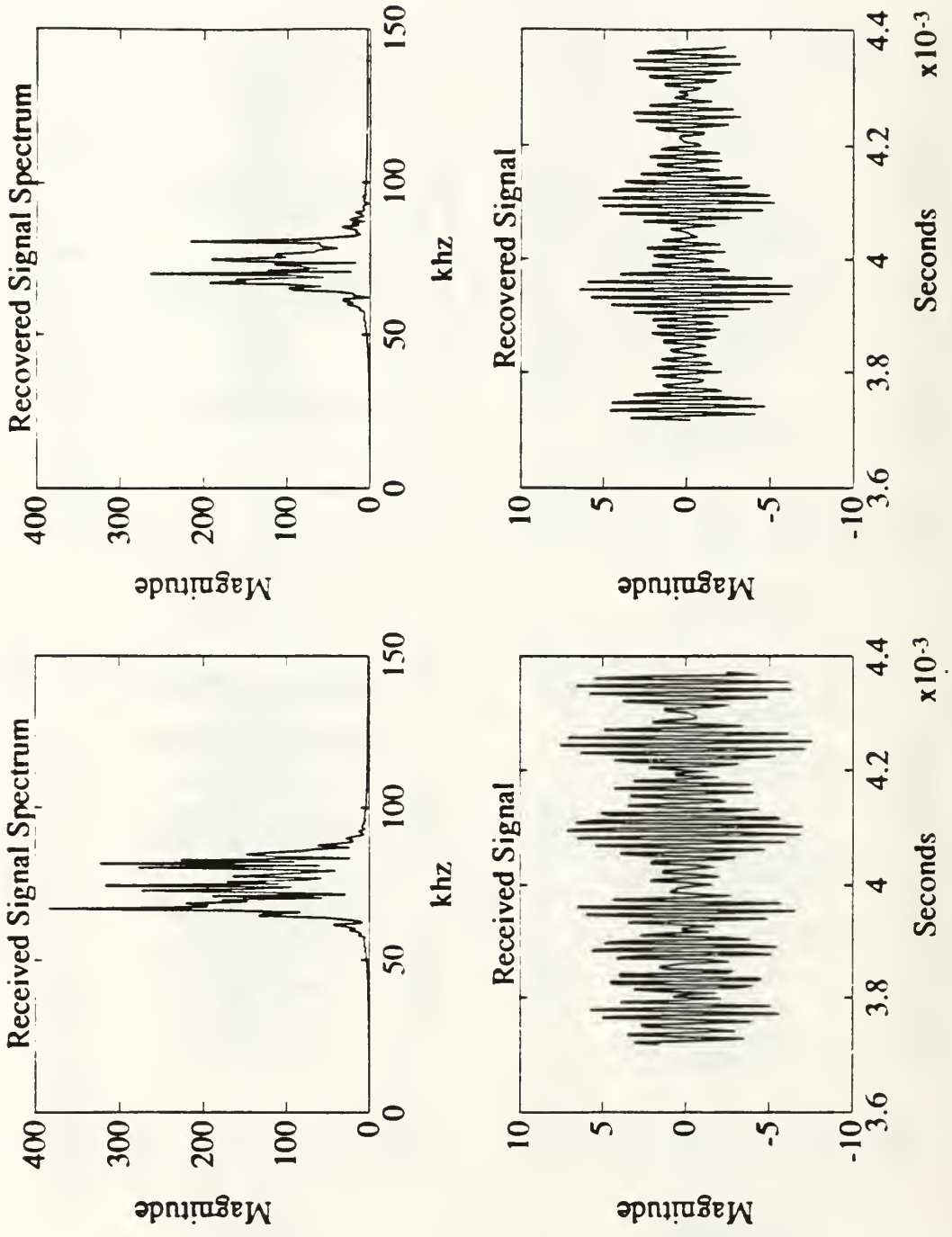
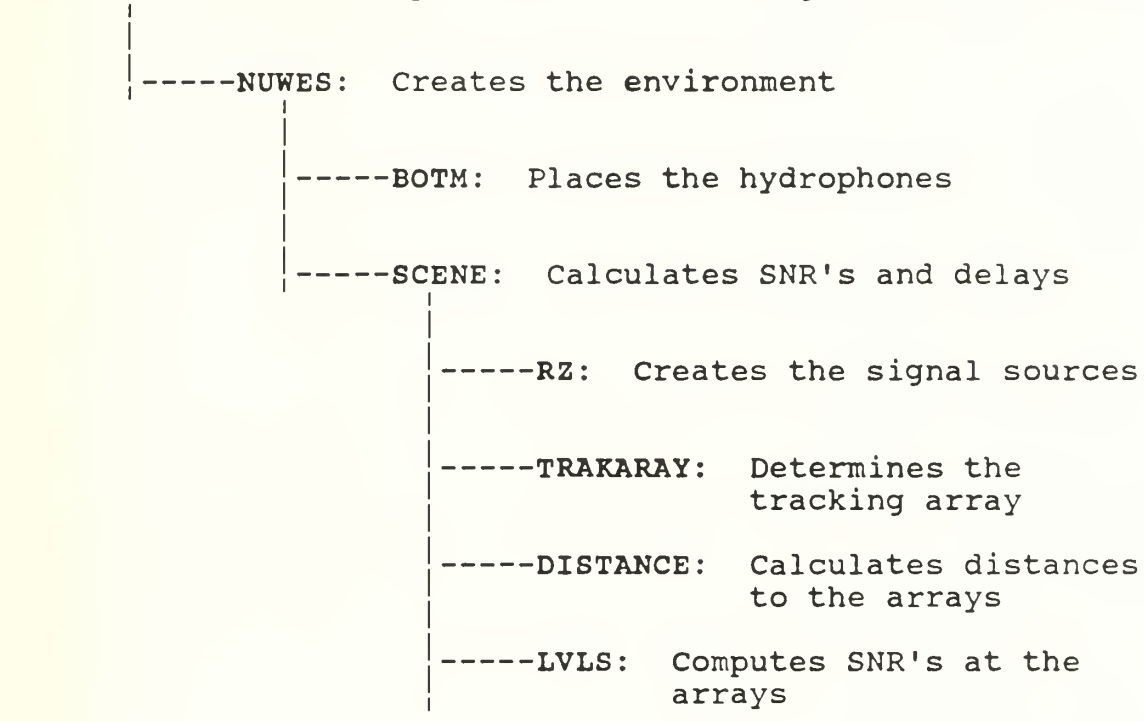


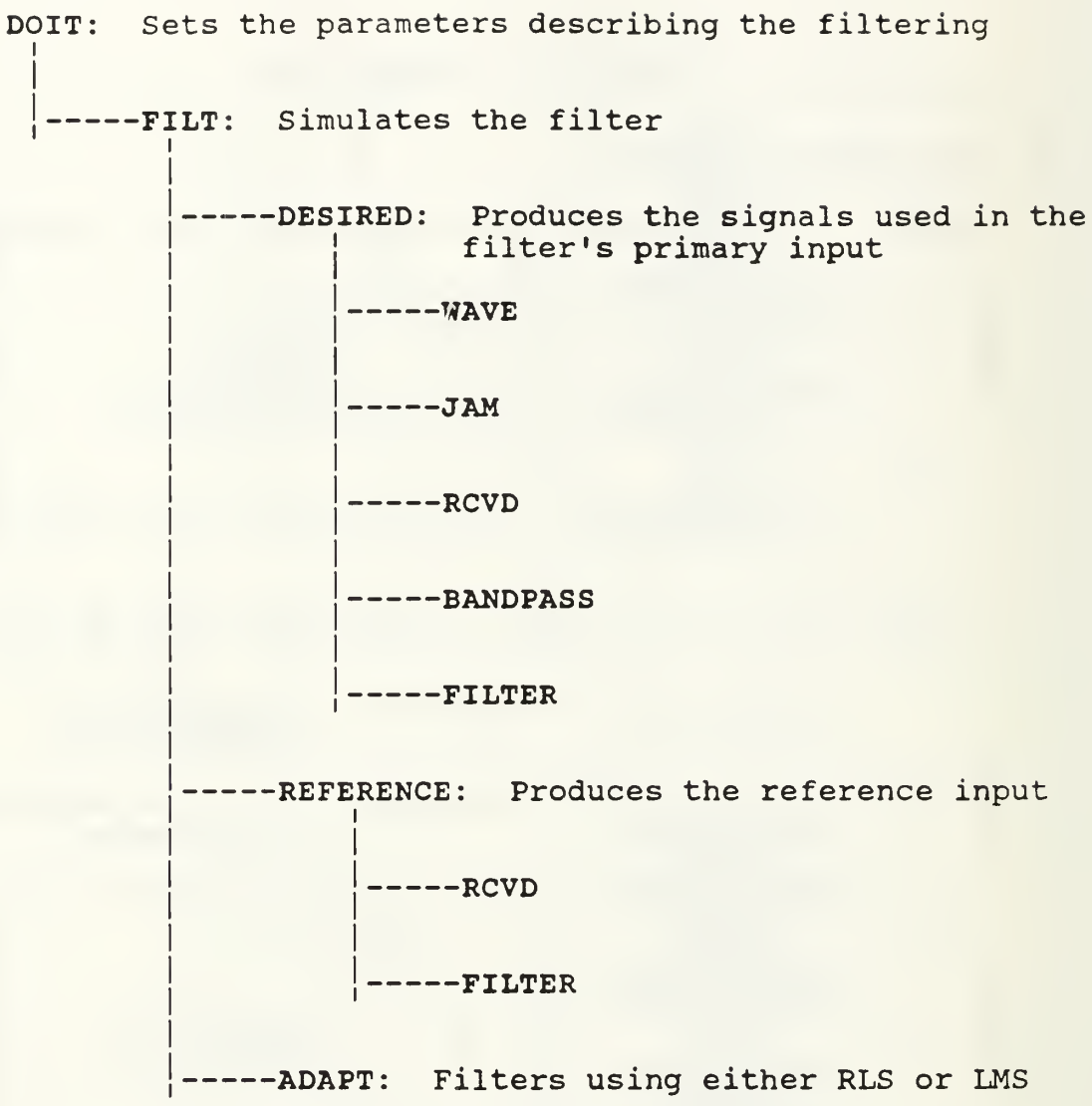
Figure 25. The received and recovered signals for an input SNR of -20 dB.

APPENDIX B. PROGRAM FLOW

A. ENVIRONMENT PROGRAMS



B. FILTERING PROGRAMS



APPENDIX C. PROGRAM LISTINGS

A. PROGRAMS THAT CREATE THE SCENARIO

1. Setup

```
% This is the main program that sets the scenarios of the
% NUWES tests
clear
% variables:
cmdpth=200;         % the depth of the countermeasure
sdpth=100;          % the depth of the signal
cmcol=35;
cmrow=20;           % the starting position for the
                    % countermeasure
scol=25;
srow=35;            % the starting position for the signal
%
%**************** call the functions ***************

[z,zs,zcm,sdelay,cmdelay,SNRS,TA,RA]=nuwes...
          (cmdpth,sdpth,cmcol,cmrow,scol,srow);

%  Returned variables:

%   z               =    the grid bottom with no signals for
%                        mesh plotting
%   zs              =    the signal source vrbl for mesh
%                        plotting
%   zcm             =    the interference vrbl for mesh
%                        plotting
%   sdelay          =    matrix of the delays for the signal
%                        source
%   cmdelay         =    matrix of the delays for the
%                        countermeasure
% **** Note *** the delays returned are relative.  The
% shortest delay in each matrix has been subtracted from all
% elements in the corresponding array.  This reduces the
% actual size of the data set required to model the system
% without changing the model.

%   SNRS is a matrix of the four following values in order
%   1 - snrdt  =  SNR for the tracking array direct path
%   2 - snrrt  =  SNR for the tracking array reflected path
%   3 - snrdr  =  SNR for the reference array direct path
%   4 - snrrr  =  SNR for the reference array reflected path
```

```
%   **** The above order is followed in all vectors used ***
%   TA          =    the chosen tracking array number
%   RA          =    the chosen reference array number

%                     ARRAY LAYOUT
%          col --> 0 ..............70
%            row  ---------------------
%             0   |   A1                |
%                 |                     |
%                 |           A3        |
%                 |                     |
%                 |   A2                |
%                 |                     |
%                 |           A4        |
%             70  ---------------------
```

2. Nuwes

```
function [z,zs,zcm,sdelay,cmdelay,SNRS,TA,RA]=nuwes...
         (cmdpth,sdpth,cmcol,cmrow,scol,srow)

% variables:
%   cmdpth          =    the depth of the countermeasure
%   sdpth           =    the depth of the signal
%   cmcol; cmrow    =    the starting position for the
%                        countermeasure
%   scol;  srow     =    the starting position for the signal
%
%   z               =    the grid bottom with no signals for
%                        mesh plotting
%   zs              =    the signal source vrbl for mesh
%                        plotting
%   zcm             =    the interference vrbl for mesh
%                        plotting
%   sdelay          =    matrix of the delays for the signal
%                        source
%   cmdelay         =    matrix of the delays for the
%                        countermeasure
% **** Note *** the delays returned are relative.  The
% shortest delay in each matrix has been subtracted from all
% elements in the corresponding array.  This reduces the
% actual size of the data set required to model the system
% without changing the model.

%   SNRS is a matrix of the four following values in order
%   1 - snrdt  =  SNR for the tracking array direct path
%   2 - snrrt  =  SNR for the tracking array reflected path
%   3 - snrdr  =  SNR for the reference array direct path
%   4 - snrrr  =  SNR for the reference array reflected path
%   **** The above order is followed in all vectors used ***
```

```
%   TA          =    the chosen tracking array number
%   RA          =    the chosen reference array number

%                     ARRAY LAYOUT
%          col --> 0 ..............70
%            row  --------------------
%             0   |   A1               |
%             .   |                    |
%             .   |            A3      |
%             .   |                    |
%             .   |   A2               |
%             .   |                    |
%             .   |            A4      |
%            70   --------------------

araydpth=400;       % the depth of the array
soundspd=1500;      % speed of sound in water is 1500 m/s
M=[-30 30];         % the mesh orientation
sdb=186;            % the signal level in dB re uPa
cmdb=sdb+20;        % the countermeasure level in dB re uPa

axis=('square');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%[X,Y,s1,s2,s3,s4,xmax,ymax,dx,dy,z]=botm;
  load bottom
% botm creates:
%           X  = the x portion of the mesh grid
%           Y  = the y portion of the mesh grid
%    s1,2,3,4  = the position of the arrays in meters as
%                measured from the top left corner (0,0)
%        xmax  = the max number of mesh intervals in the x
%                dir
%        ymax  = the man number of mesh intervals in the y
%                dir
%          dx  = the spacing in meters between intervals in
%                the x direction
%          dy  = the spacing in meters between intervals in
%                the y direction
%           z  = the botm grid with radius circles and arrays

[SNRS,cmdelay,sdelay,TA,RA,zs,zcm]=scene...
      (cmdpth,sdpth,araydpth,soundspd,sdb,cmdb,cmcol,...
      cmrow,scol,srow,X,Y,dx,dy,s1,s2,s3,s4);
```

3. Botm

```
function [X,Y,s1,s2,s3,s4,xmax,ymax,dx,dy,z]=botm
%     X  = the x portion of the mesh grid
%     Y  = the y portion of the mesh grid
```

```
%    s1,2,3,4 = the position of the arrays in meters as
%                 measured from the top left corner (0,0)
%  xmax  = the max number of mesh intervals in the x dir
%  ymax  = the man number of mesh intervals in the y dir
%    dx  = the spacing in meters between intervals in the
%          x direction
%    dy  = the spacing in meters between intervals in the
%          y direction
%     z  = the botm grid with radius circles and arrays


%  this subprogram will return the bottom configuration of
%  the NUWES test range.  The model of the bottom will
%  contain only four sensors.  This is sufficient to model
%  all desired cases of signal vs countermeasure position.
%
%
%  create the bottom grid of four offset sensor arrays with
%  a 1500 m detection radius.  Each sensor is 2500 meters
%  apart measured center to center.
%  The upper left hand corner is (0,0).

%  The positions of the sensors in (x,y) coords with x being
%  the horizontal axis are: ( in meters)

s1=[1500 1500];s2=[4000 1500];s3=[2750 3665];s4=[5250 3665];

%                        ----------------------
%                        |   s1                |
%                        |                     |
%                        |           s3        |
%                        |                     |
%                        |   s2                |
%                        |                     |
%                        |           s4        |
%                        ----------------------

%
%  Now, the mesh for the bottom will be created
%  The grid is sampled every 10 meters to decrease the
%  required points

xmax = 70; % the maximum number of samples in the x
           % direction
ymax = 70; % the maximum number of samples in the y
           % direction

range=7000; % the range length
dx=range/xmax; % the distance between x sample points
dy=range/ymax; % the distance between y sample points
```

```
%  Set up the x-y grid

xg=1:xmax;X=ones(xg)'*xg;
yg=1:ymax;Y=yg'*ones(yg);
z=zeros(xg);

%
%  now put in the rings for the sensors

for x=1:xmax
   for y=1:ymax
      if ((x*dx-s1(1))^2 + (y*dy-s1(2))^2) <= 1500^2
        z(x,y)=5;
      elseif ((x*dx-s2(1))^2 + (y*dy-s2(2))^2) <= 1500^2
        z(x,y)=5;
      elseif ((x*dx-s3(1))^2 + (y*dy-s3(2))^2) <= 1500^2
        z(x,y)=5;
      elseif ((x*dx-s4(1))^2 + (y*dy-s4(2))^2) <= 1500^2
        z(x,y)=5;
      else
        z(x,y)=0;
      end
   end
end

% put in the array markers
z(s1(1)/dx, s1(2)/dy)=10;
z(s2(1)/dx, s2(2)/dy)=10;
z(s3(1)/dx, s3(2)/dy)=10;
z(s4(1)/dx, s4(2)/dy)=10;
```

4. Scene

```
function [SNRS,cmdelay,sdelay,TA,RA,zs,zcm]=scene...
     (cmdpth,sdpth,araydpth,soundspd,sdb,cmdb,...
     cmcol,cmrow,scol,srow,X,Y,dx,dy,s1,s2,s3,s4);
% variables:
%   SNRS is a matrix of the four following values in order:
%    1 - snrdt  =  SNR for the tracking array direct path
%    2 - snrrt  =  SNR for the tracking array reflected path
%    3 - snrdr  =  SNR for the reference array direct path
%    4 - snrrr  =  SNR for the reference array reflected
%                  path
%   cmdelay     =  matrix of the delays for the
%                  countermeasure
%   sdelay      =  matrix of the delays for the signal
%                  source
%   TA          =  the chosen tracking array number
%   RA          =  the chosen reference array number
```

```
%                    ARRAY LAYOUT
%          col --> 0 ..............70
%            row  ---------------------
%              0  |   A1              |
%                 |                   |
%                 |           A3      |
%                 |                   |
%                 |   A2              |
%                 |                   |
%                 |           A4      |
%              70 ---------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%  Range and dB for countermeasure %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Rcm,zcm]=Rz(X,Y,dx,dy,cmcol,cmrow,cmdb);
%  create a noise source and place it within the bottom
%  mesh.  Then find ranges to noise source and the dB re uPa
%  level at each range using spherical spreading and linear
%  attenuation.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Range and dB for signal source %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Rs,zs]=Rz(X,Y,dx,dy,scol,srow,sdb);
%  create a signal source and place it within the bottom
%  mesh.  Then find ranges to noise source and the dB re uPa
%  level at each range using spherical spreading and linear
%  attenuation.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% find the tracking array %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The tracking array is the array closest to the signal
% source
%
[TA,RA,dsdt,dcmdt,dsdr,dcmdr]=...
      trakaray(s1,s2,s3,s4,Rs,Rcm,zs,zcm,dx,dy);
% find the array closest to the signal source and the
% non-tracking array with the lowest SJR
%   A       =    matrix of array positions
%   TA      =    the chosen tracking array number
%   RA      =    the chosen reference array number
%   dsdt    =    direct distance to tracking array from
%                the signal
%   dcmdt   =    direct distance to tracking array from
%                the countermeasure
%   dsdr    =    direct distance to reference array from
```

```
%                the signal
%   dcmdr  =     direct dist to reference array from
%                the countermeasure
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% find all of the distances and times %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% general form
%   [dtd,dtr,drd,drr]=distance(d2t,d2r,araydpth,dpth)
%   dtd        =   direct distance to tracking array
%   dtr        =   reflected distance to tracking array
%   drd        =   direct distance to reference array
%   drr        =   reflected distance to reference array
%
%   d2t        =   flat distance to tracking array
%   d2r        =   flat distance to reference array
%   araydpth   =   depth of the array
%   dpth       =   depth of the signal source
%
%   NOTE:  All depths and distances are in meters

%  find distances and times for the countermeasure

[cmdtd,cmdtr,cmdrd,cmdrr]=...
              distance(dcmdt,dcmdr,araydpth,cmdpth);
cmdist=[cmdtd,cmdtr,cmdrd,cmdrr];
cmdelay=cmdist/soundspd;

[sdtd,sdtr,sdrd,sdrr]=distance(dsdt,dsdr,araydpth,sdpth);
sdist=[sdtd,sdtr,sdrd,sdrr];
sdelay=sdist/soundspd;

% decrease the delays such that the shortest delay is zero
cmdelay=cmdelay - min(cmdelay);
sdelay=sdelay - min(sdelay);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% compute the SNR levels %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%     at the arrays      %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% general form:
%    function [snrd,snrr]=lvls(sdb,cmdb,dsd,dcmd,dsr,dcmr)
%
%  lvls computes the signal to noise levels for the
%  reflected and the direct paths to the array for which the
%  distances are given.
%
```

```
%  variables:
%       snrd    =     SNR for the direct path to the array
%       snrr    =     SNR for the reflected path to the array
%
%       sdb     =     the dB level of the signal source
%       cmdb    =     the dB level of the countermeasure source
%       dsd     =     distance from the signal direct
%       dcmd    =     distance from the countermeasure direct
%       dsr     =     distance from the signal reflected
%       dcmr    =     distance from the countermeasure
%                     reflected
%

% find the SNR for the tracking array
[snrdt,snrrt]=lvls(sdb,cmdb,sdtd,cmdtd,sdtr,cmdtr);

% find the SNR's for the reference array
[snrdr,snrrr]=lvls(sdb,cmdb,sdrd,cmdrd,sdrr,cmdrr);

SNRS=[snrdt,snrrt,snrdr,snrrr];
```

B. PROGRAMS THAT PRODUCE THE SIGNALS AND DO THE FILTERING

1. Doit.m

```
% doit
% variables
%     SNR       Signal to Noise Ratio
%     fl        filter level
%     D         the delay for the adaptive line enhancer
%     old       old = 1 then use previously calculated vrlbs
%               old = 0 create new vrbls
%     seed      the seed value for the noise generation
%     pass      pass = 1 then pass just the filter coeff and
%               inverse corr matrix to next trial (saved in
%               filtcoeff.  ie save filtcoeff a Rinv)
%     code      47 ones and zeros as desired
%     filttype  filter type
%                       0 = lms
%                       1 = rls
%     blocks    the number of blocks of data produced
%     D         the delay for the adaptive line enhancer
%                       D = 0  no ALE
%                       D > 0  use the ALE

%clear
%SNR=;      % set by start.m
%fl=;       % set by start.m
%D=;        % set by start.m
```

```
%seed=;     % set by start.m

old=0;      % if old = 1 then vrbls.mat must be the variables
            % from the previously run case.
pass=0;
%code=[ones(1,47)];
code=[1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 ...
      1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1];

filttype=1;
blocks=1;
[sn,dn,yn,en,J,seed,code,xn,f,fs,a,h,t,bits,Rinv,fl,SNR,...
tbase,bitsbase,blocks]=filt(SNR,fl,D,old,seed,pass,code,...
filttype,blocks);

save vrbls    % vrbls.mat is used when old=1

%         sn     the signal from the torpedo
%         t      the time index
%         f      the waveform frequency
%         fs     the sampling freq
%         J      the jammer signal
%         J2     the delayed jammer signal (set delay below)
%         sd     the propagated torpedo signal
%         sj     the propagated jammer signal
%         dn     the desired received signal sd+sj
%         bc,ac  the butterworth filter coeffs
%         yn     the filter output
%         en     the desired signal - the filter output
%         xn     the reference input
%         a      the adaptive filter coefficients
%         h      the computed wiener filter coefficients
%         bits   the bit index for the x-axis
%         Rinv   the correlation matrix for the adaptive
%                filter
```

2. Filt

```
f       u       n       c       t       i       o       n
[sn,dn,yn,en,J,seed,code,xn,f,fs,a,h,t,bits,Rinv,fl,SNR,...
tbase,bitsbase,blocks]=filt(SNR,fl,D,old,seed,pass,code,...
filttype,blocks)
%  this program runs the filter routine
%
%  calls:    desired        produces the desired signal
%                           and assorted variables needed
%                           throughout the program
%
%            ref            produces the reference signal
%
```

```
%             adapt          does the adaptive filtering and
%                            produces the output
%
%             taps           calculates the wiener filter tap
%                            weights for the noise assuming
%                           the noise is stationary for the %
                          sample period

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% PRINT OR NO PRINT AS THE PROGRAM PROGRESSES %%%%%%

pl=0;          % if pl = 1 then plot graphs.  match pl in the
               %  next loop down

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%% CALLING DESIRED  %%%%%%%%%%%%%%%%%%%%%%
%  DESIRED will produce the waveforms used throughout the
%          program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if pl==1
msg='entering desired.m'
end
%%%%%%%%%%%%%%%% load the old vrbls here %%%%%%%%%%%%%%%%

% Old variables are loaded when passing values from one
% program run to the next program.  Use old variables when
% you want to expand the data set w/o increasing the sample
% size.

if old==1          % old = 0 then create new values
                   % old = 1 then load the last set of vrbls
  load vrbls;
  pl=1;old=1;
  tmax=t(length(t));    % get the max time in the last
                        % iteration
  t=tbase + tmax;       % correct time for number of the
                        % last iteration
  bitmax=bits(length(bits));     % get the max bit number
  bits=bitsbase + bitmax;        % correct bit number for
                                 % iteration
  [rw,co]=size(a);
  a(1,:)=a(rw-fl,:);             % this sets the filter
                                 % coeffs to the final
                                 % values from the last use.

else                             % create new values
  [sn,t,f,fs,J,J2,sd,sj,dn,bits,bc,ac,tbase,bitsbase]=...
                        desired(SNR,code,blocks,seed);
end
```

```
        %%%%%%%%%  end of if-then %%%%%%%%%%%%
% desired creates:
%     sn        the signal from the torpedo
%     t         the time index
%     f         the waveform frequency
%     fs        the sampling freq
%     J         the jammer signal
%     J2        the delayed jammer signal (set delay in
%               desired.m)
%     sd        the propagated torpedo signal
%     sj        the propagated jammer signal
%     dn        the desired received signal sd+sj
%               after the bandpass filter
%               (if you want no BPF then go into desired.m
%               and comment
%               out the filter line and activate the line
%               below it)
%               Also, you can add a reflected jammer into dn
%     bits      the bit numbers for the x axis
%     bc,ac     butterworth filter coeffs
%     tbase     the time base
%   bitsbase    the base scale for the bits


if pl==1
msg='leaving desired.m'
end
%%%%%%%%%%%%%%%%%%%%%%%% LEAVING DESIRED %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%% CALLING REFERENCE %%%%%%%%%%%%%

if pl==1
msg='entering ref.m'
end

if old ==0
  if D==0
    [xn]=ref(J,bc,ac);    % change J to J2 when time delayed
                          % change J2 to J when no time delay
                          % set the delay in desired.m
                          % change J to J+J2 when adding a
                          % reflected jammer

%*********************************************************
% The following linea are for the adaptive line enhancer
%*********************************************************
   elseif D>0
%    [dn]=hdyne(dn,t,f);          % mult the received signal
                                  % by the local osc and put
                                  % thru LPF
```

```
     delay=D;                    % the amount of delay for
                                 % the ALE

     xn=[zeros(1,delay) dn(1:length(dn)-delay)];

  end
%*********************************************************
% end of the adaptive line enhancer variable adjustment
%*********************************************************
end

% reference creates:
%      xn       the reference signal for the adaptive filter

if pl==1
msg='leaving ref.m'
end
%%%%%%%%%%%%%%%%%%%%%%%% LEAVING REFERENCE %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%% CALLING ADAPT %%%%%%%%%%%%%%%%%%%
if pl==1
msg='calling adapt.m'
end

    %%% setup new situation data for entering adapt %%%

if old==0

% the program will enter this loop only if
% new data is being used and  the filter is adapting from
% tap weights of all zeros (initial conditions).  If the
% program does not enter this loop, then the filter will
% adapt to the data using and improving upon the old tap
% weights.

   a(1,:)=zeros(1,fl);       % start the filter coeffs with
                             % all zeros
   Rinv=200*eye(fl,fl);      % setup of the inverse
                             % correlation matx

   if pass == 1            % pass = 1 means that the filter
                           % coefficients are passed to the
                           % next array but the noise and
                           % other signals might have changed

      load filtcoeff       % retrieves 'a' and 'Rinv'

      [rw,co]=size(a);     % make the initial weights for the
```

```
                          % filter equal to the
      a(1,:)=a(rw-fl,:); % final values from the last use.
   end
end
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %%%%%%%%%% choose an adaption method %%%%%%%%%%%%%

if filttype == 0
  [en,yn,a]=adaptlms(dn,xn,fl,a);                %lms method
elseif filttype ==1
  [en,yn,a,Rinv]=adaptrls(dn,xn,fl,a,Rinv);      %rls method
end
% Creates:

%     en        the error between the desired output
%               and the filter output.  This is our
%               filtered signal
%     yn        the output of the adaptive filter
%     a         the filter coefficients.  Each row
%               is the filter coefficients as time
%               progresses.
%     Rinv      the inverse of the correlation matx. This
%               variable is passed to avoid the large
%               variations at the start of a new iteration.

save filtcoeff a Rinv    % variables used to pass parameters
                         % between arrays or between trials
if pl==1
  msg='leaving adapt'
end
%%%%%%%%%%%%%%%%%%%%%%%% LEAVING ADAPT %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%% ENTERING TAPS %%%%%%%%%%%%%%%%%%
%  Computes the wiener filter tap weights
if pl==1
  msg='entering taps'
end

%[h]=taps(sn,xn,dn,t,fl);       % h = tap weights for wiener
                                % filter assuming that the
                                % noise is stationary
                                % in the sampling period
if pl==1
   msg='leaving taps'
end
%%%%%%%%%%%%%%%%%%%%%%%% LEAVING TAPS %%%%%%%%%%%%%%%%%%
```

3. Desired

```
function [sn,t,f,fs,J,J2,sd,sj,dn,bits,bc,ac,tbase,...
         bitsbase]=desired(SNR,code,blocks,seed)
% assembles the desired signal
%  calls: wave     to produce the signal from the torpedo
%         jam      to produce the noise from the jammer
%         rcvd     to produce the received signal at the
%                  array
%         bandpass to create the filter coeffs modeling the
%                  hydrophone
%         filter   applies bandpass filter to the signals
%
%  Creates:   sn       the signal from the torpedo
%             t        the time index
%             f        the waveform frequency
%             fs       the sampling freq
%             J        the jammer signal
%             J2       the delayed jammer signal (set delay
%                      below)
%             sd       the propagated torpedo signal
%             sj       the propagated jammer signal
%             dn       the desired received signal sd+sj
%             bc,ac    the butterworth filter coeffs


[sn,t,fs,f,bits]=wavecode(code,blocks);
%produces the signal from the torpedo
%provides:
%            sn = the signal
%            t  = the time index
%            fs = the sampling freq
%          bits = the bit numbers for the x axis

%SNR in db used to scale the noise

tbase=t;        % the base set of time used for adding to
                % iterations
bitsbase=bits;  % the base set of bits used for adding to
                % iterations

K=(10^(-SNR/20)) /sqrt(2);
J=K*jam([t t],seed)/2;          % produces the noise from the
                                % jammer
J2=J(1317:1317+1315);           % produces the noise for the
                                % reference with a delay
J=J(1:1316);
[bc,ac]=bandpass(f,fs);         % produces the bandpass
                                % filter coeffs
                                % bc = the numerator coeffs
                                % ac = the denominator coeffs
```

```
path=1;
[sd]=rcvd(path,sn);      % produces received signal at the
                         % array
path=2;
[sj]=rcvd(path,J);       % produces received jammer at the
                         % array
path=3;
[sj2]=rcvd(path,J2);     % produces received reflected jammer
                         % at the array
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% produce the desired signal (comment out three of the four)
% (remember to change the reference signal in filt)
%dn=filter(bc,ac,sd+sj);        % use this line for BPF
%dn=filter(bc,ac,sd+sj+sj2);    % use this line for BPF with
                                % reflected jammer

 dn=sd+sj;                      % use this line for no BPF
%dn=sd+sj+sj2;                  % use this for reflected
                                % jammer w no BPF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% path = propagation path
% sd   = the propagated torpedo signal
% sj   = propagated jammer signal
% dn   = the combined signal + noise
%        after the bandpass filter
```

4. Ref

```
function [xn]=ref(J,bc,ac)
% assembles the reference signal
%  calls:     rcvd    to produce the received signal at the
%                     array
%
%  Plots:     none
%
%  Creates:   xn:  the signal received at the reference
%

path=3;                      % chooses the propagation path
[xn]=rcvd(path,J);
xn=filter(bc,ac,xn);
                             % produces received signal at the
reference
                             % path = propagation path
                             % J    = original jammer signal
                             % xn   = the combined signal +
                             %        noise after the bandpass
                             %        filter
```

5. Adaptlms

```
function [en,yn,a]=adaptlms(dn,xn,fl,a)
%  this function will adapt the filter parameters using the
%  Widson-Hopf algorithm (LMS)
%
%  dn  =  the desired signal
%  xn  =  the reference signal
%  fl  =  the filter order
%   u  =  acceleration parameter
%   a  =  the initial filter coefficients

% get the max length of the data vectors for loop counter

nend=length(xn);
if length(dn) > nend
   nend=length(dn);
end

% pad the data vectors with enough zeros for the loop

 xn=[xn zeros(1,nend-length(xn)+fl)];
 dn=[dn zeros(1,nend-length(dn)+fl)];


%*****  LMS METHOD  *********
% enter the loop.  the loop adjusts the filter coefficients
% with each iteration and computes error: dn-yn=en.

u=.0005;sigsq=0;

for n=1:nend
   yn(n) = a(n,:) * xn(n:n+fl-1)';    % yn for each time step
   en(n) = dn(n) - yn(n);
   delofa= -2 * en(n)*xn(n:n+fl-1);    % delta of a used to
adjust
                                       % filter coeffs in LMS
%%%%%%  The forgetting factor %%%%%%
%   sigsq=.9*sigsq + xn(n:n+fl-1)*xn(n:n+fl-1)';
%   un=u/sigsq;

   a(n+1,:)= a(n,:) - u*delofa;      % The next filter coeffs

end
```

6. Adaptrls

```
function [en,yn,a,Rinv]=adaptrls(dn,xn,fl,a,Rinv)
%  this function will adapt the filter parameters using the
%  Recursive Least Squares algorithm
```

```
%
%     dn    =    the desired signal
%     xn    =    the reference signal
%     fl    =    the filter order
%     en    =    the error signal (our output)
%     a     =    the adaptive filter tap weights


% get the max length of the data vectors for loop counter

nend=length(xn);
if length(dn) > nend
   nend=length(dn);
end

% pad the data vectors with enough zeros for the loop

 xn=[xn zeros(1,nend-length(xn)+fl)];
 dn=[dn zeros(1,nend-length(dn)+fl)];

%                *****  RLS METHOD  *********
% enter the loop.  the loop adjusts the filter coefficients
% with each iteration and computes error: dn-yn=en.

for n=1:nend

   yn(n) = a(n,:) * xn(n:n+fl-1)';   % yn for each time step
   en(n) = dn(n) - yn(n);

   k = ((Rinv * xn(n:n+fl-1)') / (1 + xn(n:n+fl-1)* ...
        Rinv*xn(n:n+fl-1)'));
   Rinv = Rinv - k*xn(n:n+fl-1)*Rinv;

   a(n+1,:) = (a(n,:)'+ k * (dn(n)-xn(n:n+fl-1)*a(n,:)') )';

end
```

LIST OF REFERENCES

Clay, C.S. and Medwin, H., Acoustical Oceanography: Principles and Applications, John Wiley & Sons, New York, 1977.

Haykin, S., Introduction to Adaptive Filters, Macmillan Publishing Company, New York, 1984.

Haykin, S., Adaptive Filter Theory, Prentice-Hall, New Jersey, 1986.

Urick, R. J., Principles of Underwater Sound, 3rd ed., McGraw-Hill, New York, 1983.

Widrow, B., and others, Adaptive Noise Cancelling: Principles and Applications, Proceedings of the IEEE vol. 63, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Murali Tummala, Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	4
5.	Professor Charles W. Therrien, Code EC/Ti Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	Dr. R. Madan (Code 1114SE) Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217-5000	1
7.	Mr. John Hager (Code 70E1) Naval Undersea Warfare Engineering Station Keyport, Washington 98345	1
8.	Captain Dale W. Herdegen 9757 Ideal Avenue North Mahtomedi, Minnesota 55115	2

Thesis
H486425 Herdegen
c.1 Adaptive noise cancellation applied to the NUWES test ranges.